PENGEMBANGAN IDS BERBASIS J48 UNTUK MENDETEKSI SERANGAN DoS PADA PERANGKAT *MIDDLEWARE* IoT

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Hilman Nihri
145150201111145

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

Pengembangan IDS Berbasis J48 Untuk Mendeteksi Serangan DoS Pada Perangkat *Middleware* IoT

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
HILMAN NIHRI
NIM: 145150201111145

Skripsi ini telah diuji dan dinyatakan lulus pada
31 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eko Sakti Pramukantoro, S.Kom, M.Kom	Ir. Primantara Hari Trisnawan, M. Sc.
NIK: 201102 860805 1 001	NIP: 19680912 199403 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 31 Juli 2018

Hilman Nihri

NIM: 145150201111145

KATA PENGANTAR

Assalamua'laikum Warahmatullahi Wabarakatuh. Syukur penulis panjatkan kehadirat Allah Subhanahu wa Ta'allah berkat rahmat dan kuasanya sehingga laporan penelitian ini bisa diselesaikan. Dalam proses pengerjaan skripsi ini, penulis memperoleh pengalaman dan ilmu. Penulis juga terbantu atas bantuan doa, dukungan baik materiil maupun non-materiil dari beberapa pihak. Oleh karena itu penulis ingin mengucapkan terima kasih kepada:

1. Orang tua dan keluarga saya yang telah memberikan doa dan dukungan untuk belajar di Universitas Brawijaya.
2. Bapak pembimbing pertama, Eko Sakti Pramukantoro, S.Kom, M.Kom yang telah memberikan bimbingan selama setahun penuh ini.
3. Bapak pembimbing kedua, Ir. Primantara Hari Trisnawan, M. Sc yang juga telah memberikan bimbingan selama setahun penuh ini.
4. Teman bimbingan, khususnya mahasiswa yang dibimbing oleh Bapak Eko Sakti Pramukantoro, S.Kom, M.Kom yang telah membantu selama proses skripsi ini berjalan.
5. Teman perkuliahan yang telah merasakan suka dan duka bersama selama penulis menempuh pendidikan di Universitas Brawijaya.
6. Dekan, Ketua Jurusan dan Ketua Program Studi Teknik Informatika.

Malang, 31 Juli 2018

Penulis

Hy.hilmann@gmail.com

ABSTRAK

Hilman Nihri, Pengembangan IDS Berbasis J48 Untuk Mendeteksi Serangan DoS Pada Perangkat *Middleware* IoT

Dosen Pembimbing: Eko Sakti Pramukantoro, S.Kom, M.Kom and Ir. Primantara Hari Trisnawan, M. Sc.

Perkembangan perangkat IoT menyebabkan perubahan pada banyak aspek kehidupan manusia. Meskipun perangkat ini memiliki keterbatasan *resource*, perangkat IoT dapat digunakan pada berbagai macam lingkungan. Penggunaan perangkat IoT ada pada lingkungan tersebut menjadikan keamanan pada perangkat IoT menjadi penting untuk dipelajari. Salah satu serangan DoS terbesar terjadi pada perangkat IoT karena tidak ada mekanisme pertahanan dini terhadap paket berbahaya sehingga perangkat IoT mudah terjangkit botnet Mirai. Metode yang dipilih pada penelitian untuk menyelesaikan permasalahan tersebut adalah dengan menggunakan *Intrusion Detection System*(IDS). IDS ini diharapkan mampu menanggulangi serangan DoS pada perangkat IoT dengan keterbatasannya. *Machine learning* dipilih sebagai pendeteksi pada IDS ini karena lebih baik dalam mendeteksi *anomaly* serta dapat berjalan lebih baik dengan keterbatasan sumber daya dibandingkan jenis IDS lainnya. Algoritma *machine learning* yang dipilih adalah J48 karena telah terbukti lebih baik dalam mendeteksi *anomaly* dibandingkan dengan algoritma klasifikasi lainnya. Terdapat beberapa parameter pengujian yang digunakan pada penelitian ini, diantaranya adalah penggunaan *resource*, akurasi *detection engine*, kemampuan memberikan *alert*, kemampuan *logging*, kehandalan dalam mengambil paket pada jaringan dan kemampuan menanggulangi serangan. Berdasarkan hasil dari pengujian tersebut, IDS ini juga mampu menanggulangi serangan, memberikan alert dan melakukan *logging*. IDS ini memiliki kemampuan mengklasifikasi paket hingga 100%, namun IDS ini hanya mampu mengambil paket pada jaringan dengan rata-rata 73,6% sehingga menyebabkan alert yang dapat ditampilkan berkisar pada rata-rata 17,42%. Penggunaan resource pada perangkat IoT meningkat dengan rata-rata penggunaan CPU sebesar 16% dan penggunaan *memory* sebesar 70MB. Berdasarkan hasil pengujian yang dilakukan, IDS dapat digunakan sebagai solusi untuk menangani serangan DoS pada perangkat IoT.

Kata kunci: *Intrusion Detection System, Internet of Thing, Security, Denial of Service, Anomaly, Machine Learning,* Weka

ABSTRACT

Hilman Nihri, Development of IDS J48-*based* To Detect DoS Attack on IoT *Middleware Device*s

Supervisors: Eko Sakti Pramukantoro, S.Kom, M.Kom and Ir. Primantara Hari Trisnawan, M. Sc.

The development of IoT devices causes a change in many aspects of human life. Although this device has a limited resources, IoT devices can be used in every kind of environment. The use of IoT device in these environments make the security of IoT device important to study. One of biggest DoS attacks happen to IoT devices because there is no self-defense mechanism toward dangerous packets, so that IoT devices easily infected by Mirai botnet. A method choosen for this research to solve this problem is using Intrution Detection System(IDS). This IDS is expected to handle DoS attack in IoT devices with its limitation. Machine learning is chosen for detector in IDS because it's better for detecting anomalies, and also can run better in limited resources than other type of IDS. The Machine Learning algorithm is J48 because J48 has been prooven to detect anomaly better than other classification algorithms. There are few testing parameters used in this research; which are resource usage, detection engine accuracy, ability to give alert, logging ability, realibility in capturing packet in the network, and ability to handle the attack. Based on the evaluation results, this IDS can handle an attack, give alert, and do the logging process. This IDS is also able to classify the packet up to 100%, but this IDS has average 73.6% for capture packet from the network, so IDS can show alert in average of 17.42%. The resource usage in this IoT devices increases by average CPU usage 16% and memory usage 70MB. Based on these testing results, IDS can be used for solution to handle DOS attack in IoT devices.

Keywords: Intrusion Detection System, Internet of Thing, Security, Denial of Service, Anomaly, *Machine Learning*, Weka.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Internet of Things(IoT) adalah sebuah teknologi baru yang memiliki peluang untuk berkembang yang tidak terbatas pada setiap aspek kehidupan manusia. Meskipun perangkat ini memiliki *resource* terbatas, perangkat IoT dapat digunakan pada lingkungan rumah sakit, reaktor nuklir dan pembangkit tenaga listrik (Cho & Choi, 2011). Dengan menggunakan IoT pekerjaan manusia menjadi lebih effisien (Ngu, et al., 2017). Dalam implementasinya, terdapat permasalahan interoperabilitas pada perangkat IoT. *Middleware* pada penelitian Anwari(2017) digunakan untuk mengatasi masalah tersebut. *Middleware* ini digunakan sebagai perangkat yang menghubungkan komputasi dan komunikasi antar protokol pada setiap perangkat (Razzaque, et al., 2016). Dengan menggunakan *middleware,* perangkat IoT tetap dapat terhubung dengan internet meskipun menggunakan protokol yang berbeda. Semua paket yang dikirimkan akan melalui *middleware* ini.

Pada bulan September tahun 2016 serangan *Distributed Denial of Service*(DDoS) terbesar terjadi dengan jumlah trafik mencapai 620 Gbps menargetkan situs dari seorang konsultan keamanan Brian Krebs, dan disaat rentang waktu yang sama pula terjadi serangan yang menargetkan situs French Webhos dan OVH dengan jumlah trafik hingga 1,1 Tbps. Serangan tersebut berasal dari perangkat IoT yang terjangkit *malware* Mirai (Kolias, et al., 2017). Ada tiga penyebab utama yang membuat serangan DoS tersebut berhasil dilakukan. Pertama dengan pertambahan jumlah perangkat IoT, tingkat pengelolaan pada jaringannya juga menurun. Kedua paket-paket berbahaya dapat dikirimkan kepada *device* IoT karena tidak semua paket dapat diperiksa satu-persatu. Ketiga *malware* yang menyebar karena kesalahan konfigurasi pada perangkat IoT (Ahmed & Kim, 2017).

Dalam mengatasi permasalah DoS pada IoT, terdapat tiga metode yang ditemukan (Kasinathan, et al., 2013). Metode pertama adalah dengan melakukan *secure bootstrapping.* Metode ini mengamankan proses perangkat ketika akan bergabung kedalam jaringan jaringan. Salah satu implementasi dari metode ini adalah dengan menggunakan WPA2-PSK. WPA2-PSK akan menyeleksi perangkat yang memiliki kata kunci yang sesuai yang dapat bergabung ke dalam jaringan. Metode ke dua adalah dengan mengimplementasikan *Aplication Layer Security*. Metode ini digunakan untuk mengamankan proses pengiriman data pada *layer* aplikasi. Salah satu contoh penggunaan metode ini adalah dengan mengunakan *Datagram Transport Layer Security* (DTLS). Dengan DTLS pengiriman paket akan menggunakan token sehingga pengiriman paket tanpa token akan didrop. Metode ke tiga adalah melakukan implementasi *Intrusion Detection System*(IDS). Metode ini akan mendeteksi paket yang berbahaya yang dikirimkan. Namun keterbatasan *memory* dan *cpu* menjadi kendala pada metode ini. Semakin banyak *rules* yang dipakai, semakin banyak *memory* dan *cpu* yang terpakai, sedangkan semakin sedikit maka kemungkinan paket lolos semakin besar (Sforzin, et al., 2016).

Berdasarkan dari metode yang ditemukan tersebut, metode penggunaan IDS memiliki keunggulan penanggulangan yang lebih baik daripada metode lainnya. Metode lainnya tidak mampu menanggulangi serangan dari DoS apabila pola dari paket yang dikirimkan diketahui oleh penyerang. Selain itu, metode implementasi IDS dipilih karena mekanisme pada metode lainnya sebagian besar hanya melindungi dasar keamanan pada jaringan seperti *confidential*, *authenticity* dan *integrity*. Sedangkan IDS adalah metode yang sudah lama digunakan dan terbukti efektif untuk menanggulangi serangan DoS(Kasinathan, et al., 2013).

Implementasi IDS pada perangkat IoT pernah dilakukan pada beberapa penelitian sebelumnya. Pada penelitian Sforzin, dkk (2016) melakukan implementasi Snort untuk menanggulangi DoS pada Raspberry Pi. Hasil penelitian membuktikan bahwa Snort dapat berjalan pada perangkat IoT. Kekurangan pada penelitian ini adalah penggunaan CPU yang mendekati *maximum usage* ketika jumlah paket per detik pada jaringan tinggi. Selain jumlah *rules* yang dapat digunakan berkisar diangka 11.000. Pengujian yang dilakukan pada penelitian ini juga hanya sebatas *node* IDS tanpa trafik dari perangkat lainnya seperti sensor. Pada penelitian Kasinathan, dkk (2013) menggunakan IDS *anomaly-based* untuk arsitektur IoT pada jaringan 6LoWPAN. IDS ini mampu mendeteksi serangan DoS dengan akurasi yang tinggi. Namun IDS ini tidak diimplementasikan pada perangkat IoT melainkan pada *computer host*. Penggunaan perangkat IoT hanya digunakan untuk menangkap paket dalam monitoring mode untuk dikirmkan kepada *computer host* IDS. Meskipun hasil yang didapatkan baik, namun penelitian ini membutuhkan *cost* yang tinggi dan tidak menyelesaikan permasalahan implementasi IDS pada perangkat dengan sumber daya terbatas.

Penelitian Kolias, dkk (2016) menggunakan algoritme *decision tree* untuk mendeteksi serangan pada jaringan 802.11. Hasil pengujian menunjukkan bahwa algoritme J48 memiliki akurasi paling tinggi diantara algoritme *decision tree* lainnya. Pada penelitian ini proses perhitungan *machine learning* menggunakan aplikasi Weka. Aplikasi Weka berisi kumpulan algoritme dari *machine learning,* salah satunya adalah J48.

Metode implementasi IDS pada jaringan IoT layak untuk dikembangkan agar implementasi IDS pada jaringan IoT dapat berjalan dengan keterbatasan perangkat IoT. IDS yang digunakan adalah *anomaly based* IDS dengan mengimplementasikan algoritme J48 sebagai *detection engine*. *Anomaly-based* IDS dipilih karena mampu mendeteksi *anomaly* dan membutuhkan *resource* yang lebih sedikit dari jenis IDS lainnya. Sedangkan algoritme J48 dipilih karena telah terbukti lebih handal dalam melakukan klasifikasi anomaly dibandingkan dengan algoritme lainnya.

1.2 Rumusan masalah

1. Bagaimana mengembangkan IDS untuk menanggulangi serangan DoS pada *middleware* IoT?
2. Bagaimana cara mengimplementasikan IDS pada *middleware* IoT?
3. Bagaiman performa IDS pada *middleware* IoT?

1.3 Tujuan

1. Mengembangkan IDS agar dapat menanggulangi DoS pada *middleware* IoT
2. Mengimplementasikan IDS pada *middleware* IoT dengan keterbatasannya
3. mengetahui performa IDS pada *middleware* IoT

1.4 Manfaat

Manfaat dari penelitian ini adalah dengan mengimplementasikan IDS pada *middleware* agar dapat meminimalisir serangan DoS dari jaringan internal sehingga perangkat IoT dapat berfungsi dengan baik. Selain itu mengetahui bagaimana cara mengimplementasikan IDS pada *middleware* IoT dan mengetahui performa dari IDS ketika diimplementasikan pada *middleware* IoT.

1.5 Batasan Masalah

Batasan masalah yang ditentukan pada penelitian ini antara lain :

a. Jenis serangan yang dideteksi pada penelitian ini adalah DoS
b. Serangan DoS yang digunakan yaitu SYN *flood* dan UDP*flood*
c. Panjang paket DoS yang dikirimkan sebesar 1024 byte
d. IDS yang dibangun berupa *anomaly-based* IDS
e. Proses perhitungan *machine learning* menggunakan aplikasi Weka

1.6 Sistematika Pembahasan

Pembahasan yang dilakukan pada penelitian ini dibagi dalam beberapa bagian, antara lain:

Bab 1 Pendahuluan Bagian ini terdiri dari latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan.

Bab 2 Landasan Kepustakaan Bagian ini membahas tentang teori perangkat, arsitektur, IDS, dan protokol pada perangkat wireless.

Bab 3 Metodologi Bagian ini berisi menjelaskan metode penelitian yang digunakan pada penelitian ini.

Bab 4 Analisis Kebutuhan Bagian ini berisi analisis kebutuhan yang dibutuhkan untuk perancangan dan implementasi dari penelitian ini.

Bab 5 Perancangan dan Implementasi Bagian ini membahas perancangan dan pengujian IDS yang dikembangkan berdasarkan analisis kebutuhan yang dilakukan sebelumnya. Bagian juga ini membahas implementasi dari perancangan sistem yang dibuat berdasarkan perancangan IDS yang dikembangkan.

Bab 6 Pengujian dan Analisis Bagian ini membahas pengujian dari implementasi IDS yang dikembangkan. Pengujian yang dilakukan meliputi pengujian kinerja IDS dan pengujian fungsional pada IDS yang telah dirancang berdasarkan perancangan pengujian yang dilakukan sebelumnya.

Bab 7 Penutupan Bagian ini membahas kesimpulan dari hasil penelitian dan saran dari penulis. Hasil penilitian akan membahas rumusan masalah yang telah ditentukan pada bagian pendahuluan.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan meliputi kajian pustaka dan dasar teori yang digunakan dalam penelitian ini. Bagian Kajian Pustaka membahas penelitian yang telah dilakukan sebelumnya yang akan digunakan pada penelitian ini. Pada bagian Dasar Teori menjelaskan tentang teori yang dikumpulkan dan digunakan dalam penelitian ini.

2.1 Kajian Pustaka

Pada bagian ini berisi referensi utama dari penelitian sebelumnya yang telah dibuat. Terdapat empat sumber utama yang akan digunakan sebagai acuan dalam penelitian ini. Tabel kajian pustaka dapat dilihat pada Tabel 2.1.

Tabel 2.1 Kajian Pustaka

No	Judul	Tahun	Penulis	Hasil	Penelitian Penulis
1	*Network* intrusion detection system using J48 Decision Tree	2015	S. Sahu, B. M. Mehtre (2015)	Mendeteksi *anomaly* trafik menggunakan algoritme J48	Menggunakan algoritme J48 sebagai *detection engine* Pada IDS
2	Anomaly *Based* Wi-Fi Intrusion Detection System	2017	Pratik Satam (2017)	Mendeteksi anomaly pada jaringan wireless	Melakukan peracangan anomaly-*based* IDS pada perangkat wireless
3	Denial-of-Service detection in 6LoWPAN *based* Internet of Things	2013	Prabhakaran Kasinathan, Claudio Pastrone, Maurizio A. Spirito , Mark Vinkovits (2013)	Melakukan *block* packet pada perangkat yang dicurigai sebagai penyerang dengan *firewall*	Penggunaan *firewall* untuk menanggulangi serangan dari attacker

2.2 Dasar Teori

Bagian ini berisi teori yang telah dikumpulan berdasarkan referensi yang ditemukan. Teori yang dikumpulkan ini akan digunakan untuk membantu penelitian ini.

2.2.1 IoT

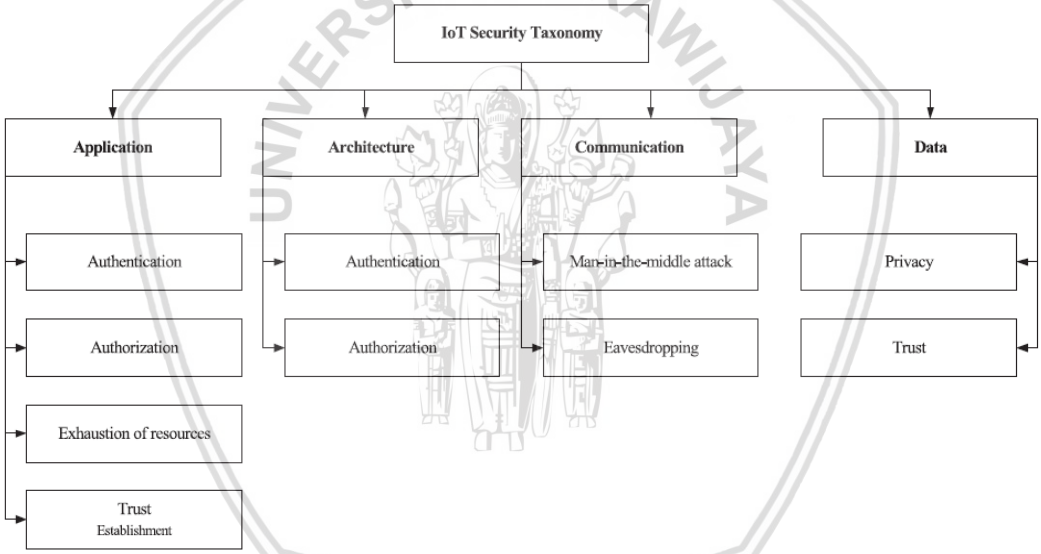
IoT adalah sebuah sistem yang teridiri dari beberapa perangkat pintar dan berkolaborasi untuk memenuhi tujuan yang sama (Sicari, et al., 2015) (Alaba, et al., 2017). Dalam beberapa tahun terakhir, lingkungan sistem IoT berubah menjadi

sistem rumit yang terdiri dari beragam sensor, *mobile device*, server, dan *embedded devices* yang memiliki teknologi berbeda (Lee , et al., 2017).

Perangkat IoT pada umumnya memiliki *resource* yang terbatas. Meskipun perangkat ini memiliki *resource* terbatas, perangkat IoT dapat digunakan pada lingkungan rumah sakit, reaktor nuklir dan pembangkit tenaga listrik. Penggunaan perangkat IoT untuk lingkungan tersebut menjadikan keamanan pada perangkat IoT menjadi penting. Oleh karena itu diperlukan sebuah mekanisme yang dapat berjalan dengan efisien dengan keterbatasan *memory*, CPU dan baterai yang dapat berjalan pada perangkat IoT (Cho & Choi, 2011).

2.2.2 Keamanan Pada Perangkat IoT

Keamanan pada IoT sendiri dibagi menjadi empat bagian. Keempat bagian ini meliputi keamanan aplikasi, arsitektur, komunikasi dan data. Penelitian ini menitik beratkan pada keamanan pada bagian aplikasi yaitu *exhaustion of resources*. Gambar 2.1 menjelaskan tentang taksonomi keamanan dari perangkat IoT (Alaba, et al., 2017).



Gambar 2.1 IoT Security Taxonomy

Sumber: Alaba, dkk., (2017)

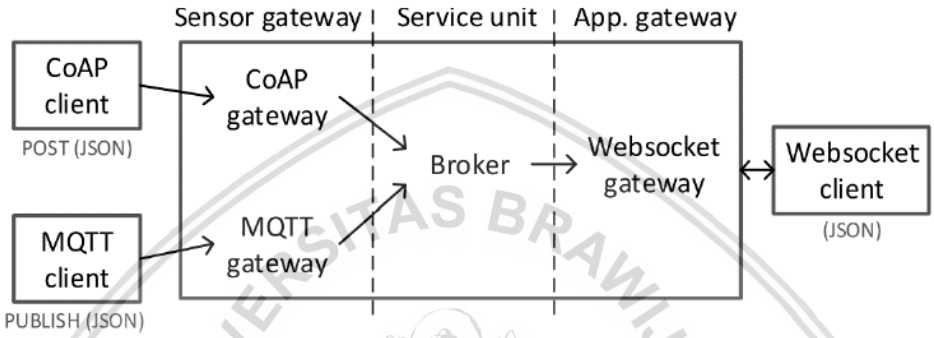
Exhaustion of resources adalah serangan yang menargetkan untuk menghabiskan *resource* sehingga dapat mengganggu layanan dari aplikasi yang berjalan pada perangkat IoT tersebut. Salah satu serangan untuk menghabiskan *resource* dapat dilakukan dengan mengirimkan packet dalam ukuran besar secara konsisten (Alaba, et al., 2017). Serangan *exhaustion of resources* merupakan salah satu tujuan dari serangan DoS (Shreenivas, 2015).

2.2.3 *Middleware* Pada IoT

Middleware pada IoT adalah perangkat yang digunakan menghubungkan komputasi dan komunikasi antar perangkat dan membantu pemrosesan

perbedaan protokol antar perangkat (Razzaque, et al., 2016). *Middleware* yang digunakan pada penelitian sebelumnya (Farizi, 2018) menggunakan paradigma *event-driven*. Pada *middleware* ini terdiri dari tiga bagian, yaitu *sensor gateway*, *service unit*, dan *application gateway*. Semua data dari sensor akan dikirim kepada *sensor gateway*.

Komunikasi pada *sensor gateway* pada penelitian ini menggunakan CoAP dan MQTT. Data yang diterima pada *sensor gateway* juga akan diteruskan kepada *service unit* yang berisi *broker* untuk diteruskan pada *subscribe*r. Untuk menjadi *subscribe*r *client* harus melakukan request melalui websocket pada app *gateway*. Arsitektur *middleware* dapat dilihat pada gambar 2.2.



Gambar 2.2 *Event Driven Middleware*

Sumber: Farizi (2018)

2.2.4 Paket Pada Arsitektur IoT

Survey pada sistem dilakukan untuk melakukan pengumpulan data. Survey meliputi proses penggalian trafik yang berjalan pada *middleware* dengan aplikasi yang tepat. Proses ini akan membantu untuk memperoleh informasi detail dari sebuah protokol. Proses ini juga merupakan salah satu dari proses untuk merancang komponen pendeteksi dari IDS (Denatious & John, 2012).

Pada penelitian ini, dibahas tentang protokol TCP, UDP, ARP, ICMP, IGMP dan EAPOL. Penjelasan dari setiap trafik dan layanan apa saja yang berjalan pada *middleware* ini dijelaskan pada bagian sub bab pada bagian ini.

2.2.4.1 TCP

Pada arsitektur penelitian sebelumnya trafik paket TCP digunakan pada beberapa service yang ditemukan pada arsitektur penelitian sebelumnya. Adapun trafik yang ditemukan adalah sebagai berikut:

a. *Secure Shell* (SSH). SSH pada umumnya digunakan untuk mengakses remote shell dengan aman, untuk mengakses file dan juga untuk membuat koneksi tunnel antar aplikasi (Gasser, et al., 2014). SSH secara umum berjalan dengan protokol TCP dengan port default 22.

b. Redis. Redis adalah aplikasi untuk menyimpan data dengan *memory*. Redis dapat menyimpan string, hash, list, set dan set yang berurutan. Untuk mengantisipasi single-point failures, Redis mereplika data dengan konsep

master-*Slave* atau yang disebut clustering pada Redis. *Master* berfungsi untuk menjaga agar data dapat masuk dan melakukan replika kepada *Slave*. Sedangkan untuk membacanya dapat melalui *slave* atau *master*. *Slave* dapat menggantikan master apabila master *crash*, dan *master* akan menjadi *Slave* sampai masalah pada master diselesaikan (Li, et al., 2017). Berdasarkan dokumentasi pada redis, redis berkomunikasi dengan protokol TCP pada port default 6379 dan port+10000 apabila berjalan dengan model cluster. Port kedua itu akan digunakan untuk bertukar data antara *node* pada Redis (Redislabs, 2018).

c. Message Queueing Telemetry Transport (MQTT). Protokol ini digunakan untuk berkomunikasi antar perangkat IoT. Protokol ini menggunakan konsep *publish subscribe*. *Publish*er akan mengirim data kepada *subscribe*r melalui *broker* (Singh, et al., 2015). Protokol ini berkomunikasi melalui protokol TCP dengan port default 1883.

2.2.4.2 UDP

Pada arsitektur penelitian sebelumnya trafik paket UDP digunakan pada beberapa service yang ditemukan pada arsitektur penelitian sebelumnya. Adapun trafik yang ditemukan adalah sebagai berikut:

a. *Dynamic Host Configuration Protocol* (DHCP). DHCP adalah modifikasi dari BOOTstrap Protocol (BOOTP). Protokol ini digunakan untuk memberikan alamat IP secara otomatis dan biasanya berjalan pada proses *boostrap*. Protokol ini menggunakan protokol UDP untuk komunikasi datanya. Perbedaan antara DHCP dan BOOTP adalah pada BOOTP alamat IP yang didapatkan tidak memiliki *lease time* dan berjalan hanya saat proses *booting*. Sedangkan DHCP menjalankan proses *discovery, offer, request* dan *acknowledge* (DORA) setiap kali lease time habis. Protokol ini berjalan pada port 67 untuk DHCP server dan 68 untuk DHCP *client* (Rajput, et al., 2016).
b. Constrained Aplication Protocol (CoAP). Protokol merupakan *application layer protocol* dan diprediksi merupakan protokol komunikasi pada aplikasi yang akan paling banyak digunakan di masa depan. Protokol ini mengadopsi prinsip dan metode dari protokol HTTP RESTful. Protokol ini digunakan banyak perangkat IoT karena ringan dan menggunakan sedikit energy (Rahman & Shah, 2016). Pada umumnya protokol ini berjalan pada port 5683.
c. *Network* Time Protocol (NTP). NTP merupakan protokol yang umum digunakan untuk menjaga sinkronisasi waktu pada jaringan. Protokol ini terbukti efektif untuk melakukan tugas tersebut (Jun, et al., 2002). Prinsip dari protokol ini adalah menghitung round-trip delay dan perbedaan waktu pada setiap mesin yang terhubung dengan internet agar memiliki waktu yang sama dengan akurasi yang tinggi (Jie, et al., 2017).

2.2.4.3 Internet Control Message Protocol (ICMP)

ICMP adalah salah satu dari protokol bawaan IP. Protokol ini didesain agar mampu melakukan *query* dan menampilkan pesan *error* untuk menguji

komunikasi antar *host* dengan *gateway* pada jaringan IP. Karena kemampuannya, protokol ini dapat dijadikan sebagai acuan dalam laporan *control management* dan *error report* (Arjuman & Manickam, 2015).

ICMP berjalan dengan protokol IPv4 dan IPv6. Pada Gambar 2.3 dan Gambar 2.4 memperlihatkan bahwa keduanya memiliki komponen yang sama yaitu *type* dan *code*. Namun pada komponen data ICMP IPv6 telah dimodifikasi untuk menyelesaikan permasalahan pada ICMP IPv4. Penggunaan ICMP IPv6 juga lebih meluas dibandingkan ICMP IPv4. Sebagai contoh fungsi protokol ARP telah dimasukkan ke dalam protokol ICMP IPv6 (Arjuman & Manickam, 2015).

0 – 7	8 – 15	16 – 23	24 – 31
Type	Code	Checksum	
Unused			
Header & 64 bits from original datagram			

Gambar 2.3 ICMP IPv4

Sumber: Arjuman, Manickam (2015)

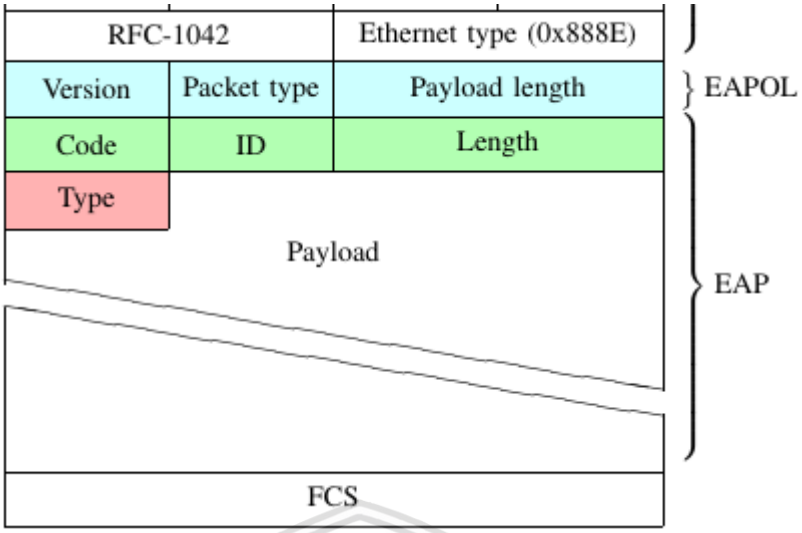
0 – 7	8 – 15	16 – 23	24 – 31
Type	Code	Checksum	
Message Body			

Gambar 2.4 ICMP IPv6

Sumber: Arjuman, Manickam (2015)

2.2.4.4 Extensible Authentication Protocol Over Local Area Networks (EAPOL)

Eapol adalah sebuah mekanisme enkapsulasi yang digunakan untuk mengirim pesan *Extensible Authentication Protocol* (EAP). EAP adalah sebuah *framework* yang digunakan untuk melakukan authentikasi pada RFC 3748. EAP tidak menjelaskan standard untuk mekanisme autentikasi, namun protokol ini menyediakan fungsi algoritme autentikasi. Pesan EAPOL akan ditransmisikan kepada *client* dan *access point* pada jaringan. Pesan ini penting karena pesan EAPOL digunakan untuk mengirimkan pesan rahasia bahkan sebelum *client* mendapatkan alamat IP (Pawlowski, et al., 2014). Gambar 2.5 menunjukkan *frame* dari protocol EAPOL.



Gambar 2.5 EAPOL Frame

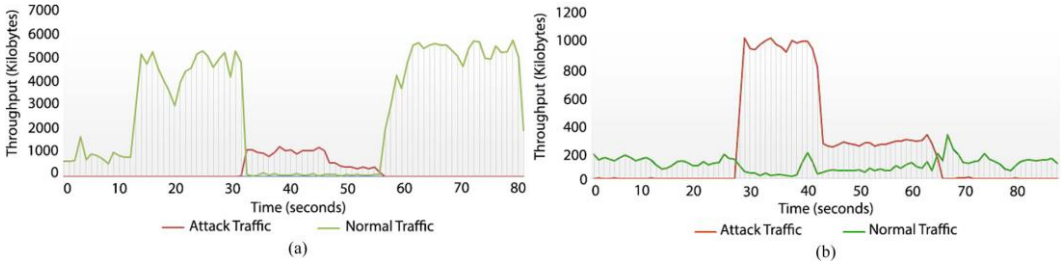
2.2.4.5 Address Resolution Protocol(ARP)

Protokol ARP digunakan untuk mencari alamat MAC dengan menggunakan alamat IPv4. Protokol ARP digunakan untuk komunikasi dua buah *host* yang berada dalam jaringan yang sama. Mereka membutuhkan alamat MAC satu sama lainnya untuk berkomunikasi. *Host* pertama akan melakukan *broadcast* pesan WHO_HAS yang berisi alamat IP tujuan. Kemudian *host* ke dua akan membalas dengan pesan ARP_REPLY yang berisi alamat MAC kepada *host* pertama. Kemudian alamat MAC yang didapatkan akan disimpan sementara waktu pada kernel dengan nam ARP *cache* sehingga kedua *host* tidak perlu mengirimkan pesan ARP setiap kali akan berkomunikasi (Prevelakis & Adi, 2017).

2.2.5 DoS

Availability adalah faktor paling penting dalam sebuah jaringan. Serangan DoS bertujuan untuk menyerang *network availability* sehingga layanan dalam jaringan tidak dapat merespon permintaan *client* dalam waktu tertentu (RGHIOUI , et al., 2014). Serangan ini biasanya terdeteksi setelah layanan jaringan tidak dapat diakses (Kasinathan, et al., 2013). Serangan DoS paling sering terjadi karena sangat mudah dilakukan. Sebaliknya, deteksi dan penanggulangannya serangan ini sangat sulit dilakukan karena serangan DoS bermacam-macam bentuknya.

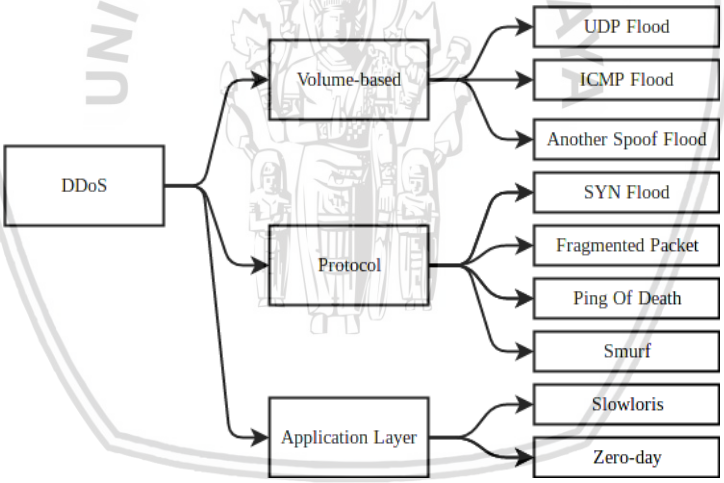
Serangan DoS pada dasarnya menargetkan *bandwidth, processor, memory,* delay komunikasi antar *node*, dan gangguan pada aplikasi, *routing* ataupun pada sistem (RGHIOUI , et al., 2014). Salah satu efek serangan DoS dapat dilhat pada Gambar 2.6, dimana serangan DoS dapat membuat *throughput* dari jaringan yang diserang turun dengan drastis.



Gambar 2.6 Gambar Drop Throughput DoS pada (a) TCP dan (b) UDP

Sumber: Kashinanta, et al. (2014)

Serangan DoS yang dilakukan secara terdistribusi disebut sebagai serangan *Distributed Denial of Service*(DDoS). Serangan ini berdasarkan penelitian (Gupta & Badve, 2017) bertujuan untuk menghabiskan sumber daya atau *bandwidth* dari korban. Serangan ini sama seperti serangan DoS hanya berbeda pada skala serangan. Terdapat tiga jenis serangan DoS pada umumnya. Jenis tersebut dapat dilihat pada Gambar 2.7. DoS *volume based* akan mengirim paket dalam ukuran besar dan banyak sehingga *bandwidth* pada jaringan penuh. Tipe serangan DoS berbasis protokol akan memanfaatkan kelemahan *flow* pada *layer transport*. Sedangkan tipe *application layer* akan menyerang kelemahan *flow* algoritme pada aplikasi yang berjalan.



Gambar 2.7 DDoS Taxonomy

Sumber: (Gupta & Badve, 2017)

Pada penelitian ini, serangan yang digunakan adalah serangan SYN *Flood* dan UDP *Flood*. Serangan ini dipilih karena komunikasi pada *middleware* dan sensor menggunakan protokol CoAP dan MQTT. Protokol ini menggunakan TCP dan UDP dalam pengiriman pesan. Sehingga serangan ini dapat dilakukan sesuai dengan lingkungan sistem yang diteliti pada penelitian ini.

Dalam pengamanan serangan DoS pada jaringan IoT menurut penelitian (Kasinathan, et al., 2013) terdapat tiga metode yang dapat digunakan untuk menanggulangi serangan DoS. Ketiga metode tersebut adalah :

1. *Secure Bootstrapping* : proses mengamankan sebuah jaringan ketika *device* akan bergabung dalam jaringan. Proses ini harus menjamin bahwa hanya perangkat yang terautentikasi yang dapat mengakses jaringan.
2. *Aplication Layer Security* : proses ini menggunakan metode keamanan dalam pengiriman pesan pada *layer* aplikasi. Salah satunya adalah *Transport Layer Secure* (TLS). Dengan menggunakan metode ini, kebutuhan dasar keamanan IoT pada dapat terpenuhi.
3. IDS : Keamanan jaringan pada umumnya menggunakan IDS sebagai langkah pertama untuk pertahanannya. Bagian utama dari IDS adalah metode pendeteksiannya. Pada umumnya metode pendeteksi pada IDS dibagi menjadi dua yaitu *signature-based* atau *anomaly-based*.

Dari ketiga metode diatas, metode yang paling baik digunakan untuk menanggulangi serangan DoS adalah metode ketiga. Metode ketiga dipilih karena mekanisme pada metode lainnya sebagian besar hanya melindungi dasar keamanan pada jaringan seperti *confidential*, *authenticity* dan *integrity*. Sedangkan IDS adalah metode yang sudah lama digunakan dan terbukti efektif untuk menanggulangi serangan DoS (Kasinathan, et al., 2013).

2.2.6 *Machine Learning*

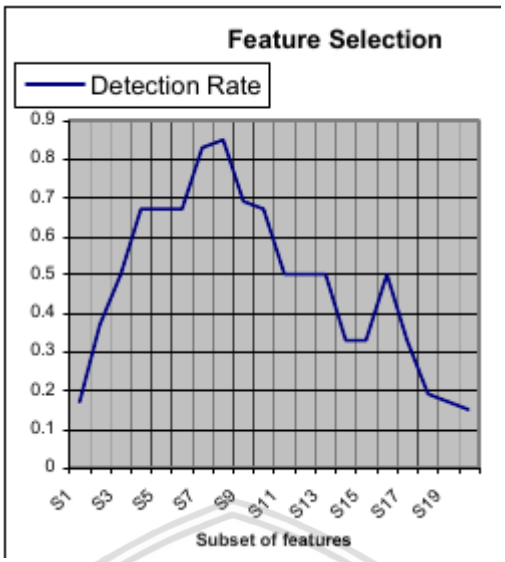
Machine Learning dapat digunakan untuk melakukan penggalian informasi pada data set yang tersedia. Dengan menggunakan perhitungan statistika dan algoritme yang matematis, *machine learning* dapat mengetahui informasi yang tersembunyi, pola dan hubungan antar atribut dalam sebuah data set. Fungsi ini menjadi sangat berguna untuk mengetahui data yang mencurigakan.

Machine learning juga dapat digunakan untuk mendeteksi serangan pada jaringan (J. & Muthukumar, 2015). Pengembangan terhadap penggunaan *machine learning* telah dikembangkan untuk mengetahui algoritme yang terbaik untuk *detection engine* pada IDS.

Machine learning yang digunakan dalam penelitian ini menggunakan *decision tree. Decision tree* digunakan karena semakin sedikit kategori data maka semakin tinggi akurasi pada metode ini (Sahu & Mehtre, 2015). Berdasarkan pengumpulan data sebelumnya, penelitian ini sesuai dengan metode ini karena jaringan IoT pada penelitian ini tidak memiliki banyak kategori trafik.

Pada dasarnya data yang dikumpulkan untuk *machine learning* berukuran sangat besar. Sehingga perlu dilakukan pengurangan pada data untuk mengurangi sumber daya yang dibutuhkan. Pengurangan dapat menggunakan metode filter, clustering atau penyeleksian fitur. (Gul & Adali, 2017)

Penyeleksian fitur adalah proses untuk meminimalisir data dengan mengeliminasi fitur yang redundant atau tidak relevan. Proses ini penting karena dapat mempengaruhi hasil akurasi dari *machine learning.* Perbandingan akurasi pada penelitian sebelumnya dapat dilihat pada Gambar 2.8.



Gambar 2.8 Grafik Akurasi Pemilihan Fitur Berdasarkan Jumlah Fitur

Sumber: Gul & Adali (2017)

Terdapat tiga metode untuk melakukan penyeleksian fitur. Ketiga metode tersebut adalah *filter, wrapper* dan *embedded*. Metode filter dilakukan dengan menghapus fungsi yang memiliki nilai rendah ketika menghitung nilai dari setiap fitur. Metode *wrapper* akan mencoba kombinasi yang berbeda dalam setiap fitur menggunakan data mining dan memberikan hasil fitur terbaik. Metode embedded atau hybrid adalah gabungan dari filter dan *wrapper*. (Gul & Adali, 2017).

Pada penelitian ini, model yang digunakan untuk pemilihan fitur adalah model hybrid. Model ini akan menghitung nilai dari setiap fitur dan melakukan percobaan dengan memilih fitur dengan nilai tertinggi untuk mendapatkan data subset yang optimal. Algoritme yang digunakan adalah *Information Gain Ratio*(IGR). Algoritme ini mampu menghitung fitur yang paling berpengaruh pada *decision tree* yang akan dibuat.

Proses perhitungan IGR dimulai dari perhitungan *entropy* dari *datasaet. Entropy* digunakan mengukur ketidakpastian suatu variabel acak. Semakin tinggi nilai *entropy*nya semakin tinggi kemungkinan fitur tersebut digunakan sebagai *root node* pada *decision tree*. Persamaan 2.1 adalah rumus dari perhitungan *entropy*.

$$Entropy(Ex) = -\sum P_i log_2(P_i) \quad (2.1)$$

Setelah nilai setiap *entropy* dari fitur diketahui, kemudian akan dilakukan perhitungan *Information Gain*(IG). Rumus perhitungan dari IG dapat dilihat pada Persamaan 2.2

$$IG(Ex, f) = Entropy(Ex) - \sum_{v \in Values(f)} \frac{|Ex,v|}{|Ex|} \times Entropy(Ex, v) \quad (2.2)$$

Setelah nilai setiap *entropy* dari fitur diketahui, kemudian akan dilakukan perhitungan IGR. Rumus perhitungan dari IG dapat dilihat pada Persamaan 2.4.

Pada persamaan 2.4, nilai dari IG akan dinormalisasi dengan Persamaan 2.3 sehingga akurasi IGR meningkat.

$$Entropy(Ex) = -\sum_{V \in Values(f)} \frac{|Ex,v|}{|Ex|} log_2(\frac{|Ex,v|}{|Ex|}) \quad (2.3)$$

$$IGR(Ex,f) = \frac{IG(Ex,f)}{SplitInfo(Ex,f)} \quad (2.4)$$

2.2.7 J48

Setelah pemilihan fitur selesai dilakukan, maka akan dilakukan pengujian dengan data training. Dengan melakukan pengujian pada data yang telah dipilih fiturnya, waktu pemrosesan dan juga akurasi hasil dari *machine learning* juga berubah. Sehingga akurasi data dengan fitur yang optimal akan didapatkan.

Algoritme J48 merupakan algoritme yang mengimplementasikan algoritme C4.5 dalam bahasa java untuk aplikasi Weka. Algoritme ini dibuat pada tahun 1993 oleh Quinalan. Algoritme ini termasuk dalam metode klasifikasi dengan berbasis *divide and conquer*. Sebuah decision tree terdiri dari *node* dan *leaf node*s. *Node* akan melakukan *test* pada attribute dan *leaf* adalah kelas dari klasifikasi. Setiap path pada hasil J48 adalah rule yang akan digunakan pada IDS. Pseudo code algoritme J48 dapat dilihat pada Gambar 2.9

Algorithm 1 Pseudo code for C4.5 (J48) algorithm

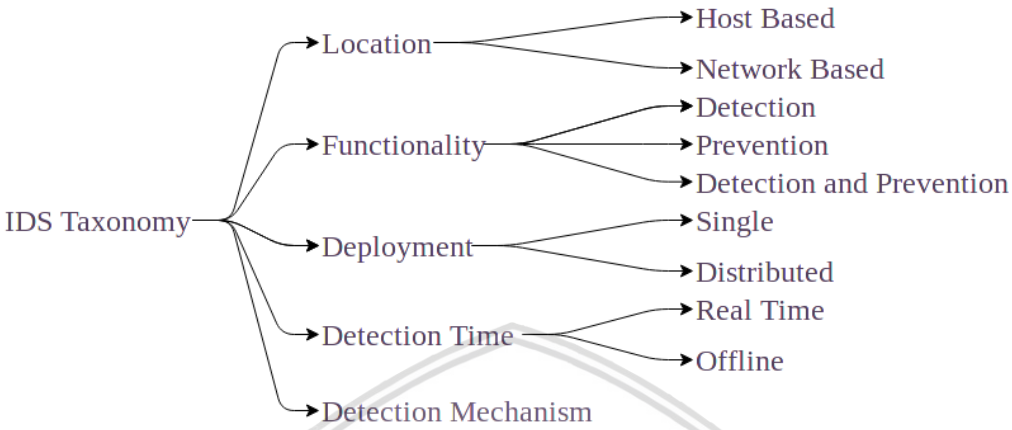
```
1: Create a root node N;
2: If (T belongs to same category C)
       { leaf node=N;
       mark N as class C;
       return N; }
3: For i=1 to n
       { Calculate Information_gain(A_i);}
4: t_a= testing attribute;
5: N.t_a= attribute having highest information_gain;
6: if (N.t_a==continuous)
       { find threshold; }
7: For (Each T' in the splitting of T)
8:         if (T' is empty)
               { child of N is a leaf node;}
9:         else
               {child of N= dtree (T')}
10: calculate classification error rate of node N;
11: return N;
```

Gambar 2.9 *Pseudocode* J48

2.2.8 IDS

Peningkatan perkembangan jaringan, kecepatan pertukaran data dan penggunaan internet yang tidak dapat diprediksi dapat menambah masalah yang mengakibatkan kegagalan pada sistem. Sehingga banyak pengembangkan sistem untuk mengurangi permasalahan-permasalahan ini. Sistem yang reliable, efektif, dan dapat memonitor serta dapat bertindak secara otomatis tanpa campur tangan

manusia adalah sistem yang dikembangkan untuk menyelesaikan permasalahan-permasalahan ini. IDS adalah salah satu sistem yang dikembangkan untuk mendeteksi serangan berdasarkan informasi yang diperoleh dari merekam trafik pada jaringan (J. & Muthukumar, 2015).



Gambar 2.10 IDS Taxonomy

Sumber: Phrate, dkk (2014)

Taksonomi pada IDS dibagi menjadi lima bagian. Setiap bagian tersebut harus diperhitungkan berdasarkan tujuan penggunaannya dan keuntungan kerugiannya. Kelima hal tersebut adalah lokasi, fungsi, penyebarannya, waktu pendeteksiannya dan mekanisme pendeteksiannya (Pharate, et al., 2015). Gambar 2.10 menjelaskan tentang taksonomi dari IDS.

2.2.8.1 Lokasi IDS

Lokasi penempatan IDS dapat menggunakan *host based* ataupun *network based*. *Host based* pada umumnya lebih mudah digunakan dan tidak membutuhkan *bandwidth* yang lebih banyak, tetapi tidak semua jenis serangan dideteksi dan IDS jenis ini tidak dapat menanggulanginya secara otomatis. Bahkan IDS jenis ini dapat dijadikan target untuk mematikan sistem IDS pada jaringan (Pharate, et al., 2015).

Network based pada umumnya lebih tinggi tingkat kegagalannya dan membutuhkan *bandwidth* sebesar jaringan yang dilindungi. Namun jenis ini lebih adaptive dan dapat mendeteksi lebih banyak jenis serangan (Pharate, et al., 2015).

2.2.8.2 Fungsionalitas IDS

Fungsi IDS sendiri adalah sistem tersebut memiliki fungsionalitas untuk mendeteksi, ataupun hanya untuk tindakan penanggulangan saja ataupun digunakan untuk keduanya (Pharate, et al., 2015).

IDS jenis *detection* hanya mampu mendeteksi serangan tanpa melakukan tindakan *preventive*, Sedangkan IDS jenis *proactive* dapat melakukan tindakan penanggulangan otomatis dan memiliki akses data terhadap *layer* dua sampai tujuh pada OSI *layer* sehingga dapat mencegah serangan lebih awal. IDS jenis kedua ini sering disebut sebagai *Intrusion Prevention System*(IPS). IDS jenis ketiga

adalah gabungan dari IDS dan IPS. Jenis ini digunakan apabila ada serangan yang dapat melewati IPS sehingga dibutuhkan IDS tambahan (Pharate, et al., 2015).

2.2.8.3 Penempatan IDS

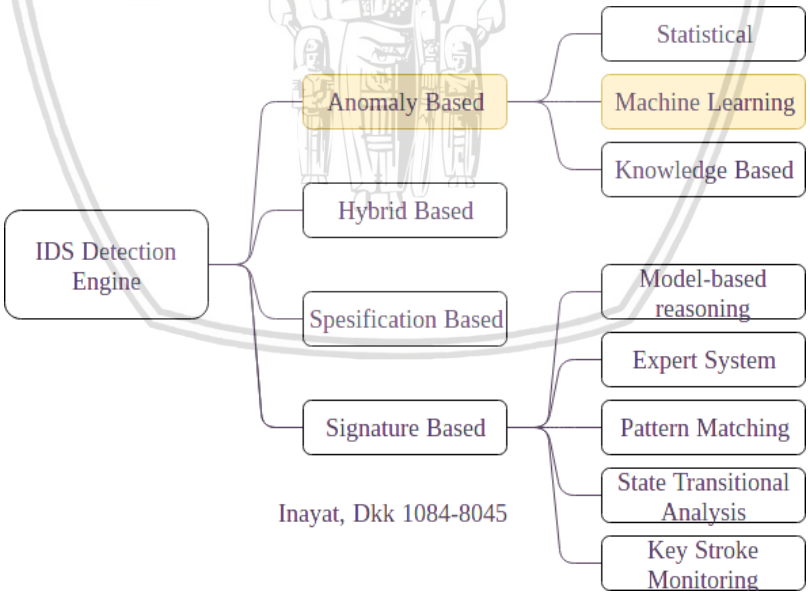
Penempatan IDS dibagi menjadi dua. Pertama ada jenis *single host* digunakan untuk jaringan yang tidak terlalu besar dan trafik yang rendah. Kedua adalah *distributed* untuk jaringan yang luas dan memiliki trafik yang tinggi sehingga IDS bekerja secara terdistribusi untuk meningkatkan kemampuan dalam mendeteksi serangan (Pharate, et al., 2015).

2.2.8.4 Waktu Pendeteksian IDS

Mekanisme pendeteksian juga dibagi menjadi dua, yaitu secara *real time* ataupun *offline* (Pharate, et al., 2015). Penggunaan *real time* memberikan keuntungan bahwa serangan dapat dideteksi pada saat serangan terjadi sehingga dapat ditangani secara langsung. Sedangkan penggunaan *offline* dilakukan pemrosesan untuk mendeteksi serangan terhadapt trafik yang disimpan dalam sebuah *database*.

2.2.8.5 Mekanisme Pendeteksi IDS

IDS memiliki empat jenis *detection machine* untuk mendeteksi serangan. Setiap pendeteksi tersebut memiliki keunggulan dan kekurangan masing-masing. Keunggulan dan kekurangan ini akan menentukan keberhasilan dari implementasi IDS (Inayat , et al., 2017). Keempat jenis tersebut dapat dilihat pada Gambar 2.11.



Gambar 2.11 IDS Detection Engine

Sumber: Inayat, dkk (2015)

Signature-based memiliki kelemahan pada penggunaan *resource* yang besar untuk menyimpan *rules* yang akan digunakan. Semakin banyak *rules* yang dipakai maka semakin besar *resource* yang akan digunakan, namun semakin tinggi kemungkinan untuk mendeteksi serangan (RGHIOUI , et al., 2014). Selain

penggunaan *memory* yang semakin besar, pada IDS jenis ini juga akan menghabiskan *resource* untuk melakukan *signature matching* pada paket yang diterma (Pongle & Chavan, 2015).

A*nomaly-based* memiliki kelemahan nilai *false positif* yang tinggi namun membutuhkan lebih sedikit *memory* (RGHIOUI , et al., 2014). Jenis ini lebih ringan karena proses pembuatan dilakukan diluar sistem IDS. Sehingga *rules* yang diperoleh dapat digunakan untuk mendeteksi *anomaly* pada packet yang diterima.

Salah satu metode klasifikasi pada *machine learning* adalah *decision tree*. Terdapat beberapa algoritme *decision tree* yang dapat digunakan pada IDS. Pada penelitian sebelumnya, telah dievaluasi penggunaan algoritme *decision tree* yang telah ada. Hasil akurasi terbaik yang didapatkan untuk mendeteksi serangan ada pada algoritme J48. Hasil tersebut ditunjukan pada Tabel 2.2.

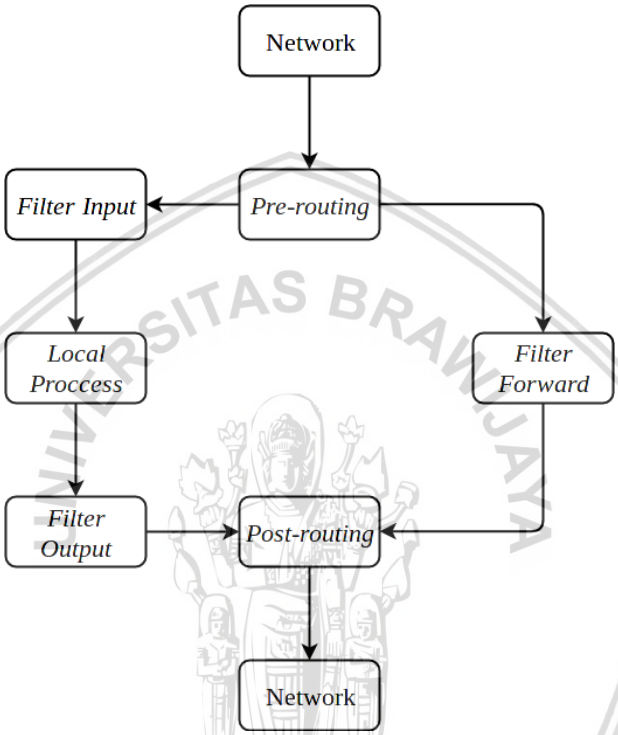
Tabel 2.2 Perbandingan Kinerja Algoritme Klasifikasi

Algorithm	Correctly %	Incorrectly %	TP Rate	FP Rate	Precision
Adaboost	92.2073	7.7927	0.922	0.922	0.85
Hyperpipes	92.2363	7.7637	0.922	0.919	0.879
J48	***96.2574***	***3.7426***	***0.963***	***0.436***	***0.962***
Naïve Bayes	90.5504	9.4496	0.906	0.399	0.917
OneR	94.5741	5.4259	0.946	0.642	0.9
Random Forest	35.8247	4.1753	0.958	0.493	0.959
Random Tree	96.2258	3.7742	0.962	0.438	0.959
ZeroR	92.2073	7.7927	0.922	0.922	0.85

Sumber: (Kolias, et al., 2016)

2.2.9 *Firewall*

Firewall sendiri memiliki mekanisme dalam mengatur paket yang melaluinya. *Firewall* dapat melakukan drop paket pada bagian filter *input*, filter *forward* dan filter *output*. Filter *input* akan menghentikan paket dari proses pada *layer network*, filter *forward* akan menghentikan paket untuk diteruskan, sedangkan filter *output* akan menghentikan paket yang akan dikirim keluar. Penelitian ini akan menggunakan ketiga *rules* tersebut. *Flow* dari *firewall* dapat dilihat pada Gambar 2.12.



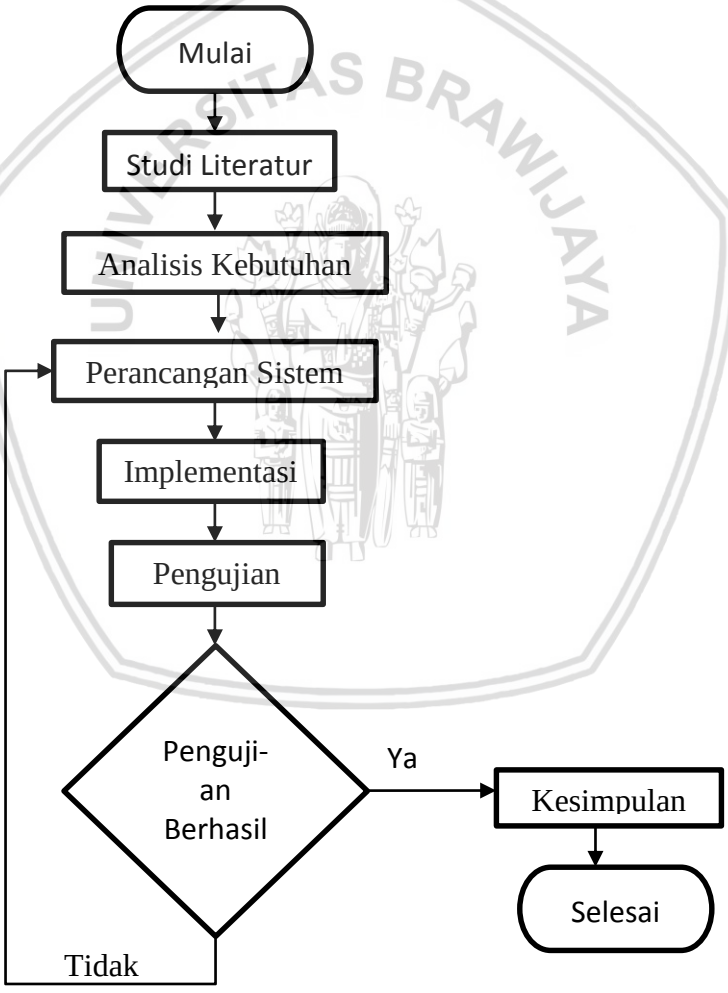
Gambar 2.12 *Firewall Flow* Pada *Iptables*

BAB 3 METODOLOGI

Bagian ini akan dibahas mengenai metode penelitian yang digunakan dalam penelitian ini. Pada bagian jenis penelitian terdapat diagram alir proses penelitian yang menjelaskan proses alur dari penelitian ini. Bagian metodologi penelitian berisi penjelasan dari setiap alur dari penelitian ini.

3.1 Jenis Penelitian

Penelitian ini menggunakan diagram alir *top-down* sesuai Gambar 3.1. Penelitian ini akan dimulai dengan studi literatur hingga ke bagian kesimpulan. Keberhasilan hasil pengujian akan menentukan proses untuk melanjutkan ke kesimpulan atau kembali ke perancangan hingga pengujian tersebut berhasil. Jenis penelitian pada penelitian ini adalah implementatif pengembangan.



Gambar 3.1 Diagram Alir Penelitian

3.2 Metodologi Penelitian

Bagian ini menjelaskan setiap fungsi dari bagan diagram alir penelitian pada Gambar 3.1.

3.2.1 Studi Literatur

Studi literatur adalah proses untuk mengumpulkan teori berdasarkan penelitian sebelumnya untuk digunakan pada penelitian ini. Tujuan dari studi literatur adalah untuk menggali informasi dari pustaka yang berkaitan dengan penelitian, serta membandingkan dengan penelitian yang telah dibuat sebelumnya. Adapun teori yang dipelajari yaitu:

a. Taksonomi IDS
b. *Machine Learning* J48
c. IDS pada perangkat IoT

3.2.2 Analisis Kebutuhan

Analisis kebutuhan adalah proses penggalian kebutuhan untuk perancangan dan implementasi pada penelitian ini. Pada bagian ini proses penggalian kebutuhan menggunakan konsep *Network Development Life Cycle.* Konsep ini akan membantu untuk merancang dan mengimplementasikan penelitian ini sesuai dengan teori yang telah dikumpulkan sebelumnya. Analisis kebutuhan meliputi kebutuhan sistem, kebutuhan jaringan, kebutuhan *machine learning* J48 dan kebutuhan aplikasi IDS.

3.2.3 Perancangan Dan Implementasi

Bagian ini memiliki perancangan jaringan dan perancangan aplikasi. Perancangan jaringan meliputi perancangan arsitektur jaringan dan perancangan aliran data. Perancangan aplikasi meliputi perancangan *use case* dan *sequences diagram*. Implementasi merupakan penerapan dari perancangan yang telah dibuat sebelumnya.

3.2.4 Pengujian dan Analisis

Proses pengujian dilakukan sesuai dengan metrik pengujian yang telah ditentukan pada bagian perancangan. Setelah proses pengujian selesai akan dilakukan analisis apakah pengujian tersebut berhasil atau gagal berdasarkan rumusan masalah yang telah dibuat. Apabila terdapat kegagalan maka akan kembali pada proses perancangan. Bagian ini adalah bagian yang akan menentukan keberhasilan dari penelitian ini.

3.2.5 Kesimpulan

Proses pengambilan kesimpulan didapatkan dari hasil analisa pengujian yang telah dilakukan sebelumnya.

BAB 4 ANALISIS KEBUTUHAN

4.1 Deskripsi Umum Sistem

Sistem yang dibuat merupakan pengembangan dari penelitian sebelumnya. Pada sistem ini terdapat *middleware* berfungsi sebagai penghubung antara sensor dan *gateway device*. Semua data dari sensor ataupun *gateway device* akan dikirim melewati *middleware*. Oleh karena itu *middleware* sangat penting dalam arsitektur in. Tujuan dari penelitian ini adalah agar pada sistem ini dapat menanggulangi serangan DoS pada *middleware* sehingga proses pada sistem ini dapat tetap berjalan. Proses yang dilakukan untuk melindungi *middleware* meliputi proses pengambilan paket, proses deteksi serangan, dan proses penanggulangan.

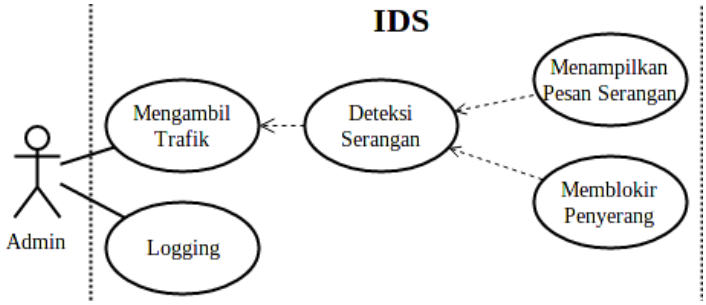
Berdasarkan penilitian sebelumnya, solusi yang dipilih pada penelitian ini adalah dengan mengimplementasikan IDS pada perangkat IoT. IDS ini dibuat dengan memperhatikan taksonomi dari IDS. IDS ini akan dipasang pada *middleware* pada sistem ini. Dengan pemasangan IDS, *middleware* tersebut diharapkan dapat menanggulangi serangan DoS dan trafik normal seperti MQTT, CoAP, Redis, DHCP, NTP ARP, EAPOL, IGMP, DNS, MDNS, SSH, ICMP tidak terganggu.

4.2 Kebutuhan Sistem

Pada bagian ini akan dijelaskan tentang kebutuhan fungsional dan non-fungsional serta kebutuhan perangkat lunak dan keras. Kebutuhan fungsional dan non-fungsional merupakan kebutuhan yang diperlukan pada sistem yang akan dibuat. Kebutuhan perangkat lunak dan perangkat keras adalah kebutuhan yang dapat menunjang keberhasilan penilitian ini.

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan fungsi yang harus dipenuhi agar aplikasi IDS dapat berjalan sesuai tujuan. Pada penilitian ini diperlukan fitur mengambil trafik jaringan, deteksi serangan, memblokir penyerang, menampilkan pesan serangan dan *logging*. Gambar 4.1 menunjukkan *usecase diagram* dari aplikasi IDS pada penelitian ini.



Gambar 4.1 *Use Case* Aplikasi IDS

Usecase diagram terdiri dari *usecase* dan *actor*. Diagram ini dapat membantu mempresentasikan hubungan antara jenis pengguna dan fitur ke dalam bentuk gambar. Setiap fitur akan direpresentasikan ke dalam bentuk sebuah *use case* dan setiap jenis pengguna akan direpresentasikan ke dalam bentuk *actor*. Penjelasan dari setiap *use case* dapat dilihat pada Tabel 4.1.

Tabel 4.1 Kebutuhan Fungsional

No	Kebutuhan Fungsional	Penjelasan
1	Mengambil Trafik Jaringan	Sistem mampu mengambil trafik dari jaringan yang akan dilindunginya
2	Deteksi Serangan	Sistem mampu mendeteksi serangan SYN *flood* dan UDP *flood* yang terjadi dengan melakukan seleksi menggunakan *machine learning j48* dari paket yang diambil dari jaringan
3	Memblokir Penyerang	Sistem mampu menanggulangi serangan dengan memblokir paket dari penyerangnya setelah terdeteksi serangan DoS
4	Menampilkan *alert*	Sistem dapat menampilkan peringatan pesan serangan pada *console* ketika terdeteksi serangan DoS
5	Menyimpan Log	Sistem melakukan penyimpanan aktifitasnya ke dalam sebuah file log

4.2.2 Kebutuhan Non-Fungsional Aplikasi IDS

Kebutuhan Non-fungsional adalah kebutuhan tambahan yang dapat meningkatkan keberhasilan tujuan apabila dipenuhi. Kebutuhan non-fungsional pada aplikasi IDS ini terdapat dua, yaitu kebutuhan *availability* dan *performance*. Kebutuhan ini dapat dilihat pada Tabel 4.2.

Tabel 4.2 Kebutuhan Non-Fungsional

No	Kebutuhan Non-Fungsional	Penjelasan
1	*Availability*	1. Sistem dapat berjalan 24/7 baik ketika ada serangan ataupun tidak ada serangan 2. Sistem dapat melakukan *restart* ketika *crash*
2	*Performance*	Sistem dapat melakukan klasifikasi paket dibawah satu detik

4.2.3 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak adalah kebutuhan aplikasi yang digunakan pada penelitian ini untuk menunjang keberhasilan dari aplikasi IDS pada penelitian ini. Adapun kebutuhan perangkat lunak dapat dilihat pada Tabel 4.3. dan Tabel 4.4.

Tabel 4.3 Kebutuhan Perangkat Lunak

No	Aplikasi	Fungsi
1	*Iptable*s	Aplikasi ini berfungsi untuk menggunakan fitur *firewall* pada kernel. Aplikasi ini akan digunakan untuk membuat rule pada *firewall*

Tabel 4.4 Kebutuhan Perangkat Lunak Lanjutan

No	Aplikasi	Fungsi
2	Python-*iptable*s	Python-*iptable*s adalah *package* pada python yang akan digunakan untuk membuat *rules* pada *firewall*. *Library* ini akan mengeksekusi dari program python ke *iptable*s.
3	Scapy-python3	Scapy merupakan *library package* dari python yang digunakan untuk memanipulasi paket. Scapy juga mampu membaca file binary paket dan mengambil paket secara langsung pada jaringan.
4	Weka	Aplikasi ini merupakan aplikasi komputasi cerdas yang digunkan untuk memroses data. Pada sistem ini akan menggunakan algoritme IGR dan J48 pada aplikasi ini.
5	Tcpdump	Aplikasi berbasis console yang digunakan untuk membaca atau merekam paket pada jaringan.
6	Atop	Aplikasi yang digunakan untuk memonitor *resource* mesin seperti CPU, *memory*, swap dan *bandwitdth*

4.2.4 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras adalah kebutuhan perangkat yang akan digunakan untuk menunjang penelitian ini. Kebutuhan ini dapat dilihat pada Tabel 4.5.

Tabel 4.5 Kebutuhan Perangkat Keras

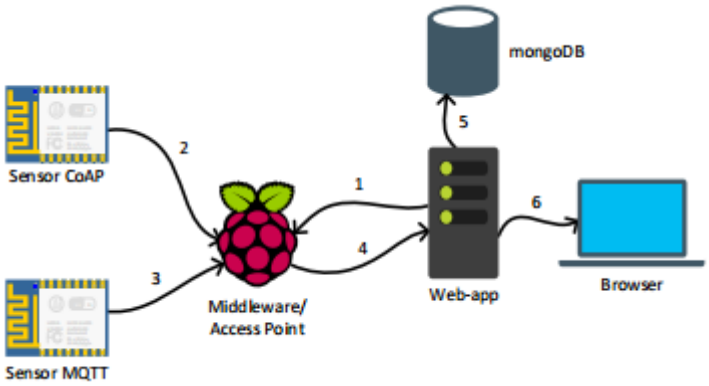
No	Perangkat	Fungsi
1	Raspberry-pi	Raspberry akan digunakan sebagai tempat implementasi ids dan penyerangan

4.3 Kebutuhan Jaringan

Perancangan jaringan adalah rancangan jaringan dari sistem yang digunakan dalam penelitian ini. Rancangan ini meliputi lingkungan penelitian sebelumnya, alur komunikasi dan topologi yang digunakan pada sistem.

4.3.1 Lingkungan Penelitian Sebelumnya

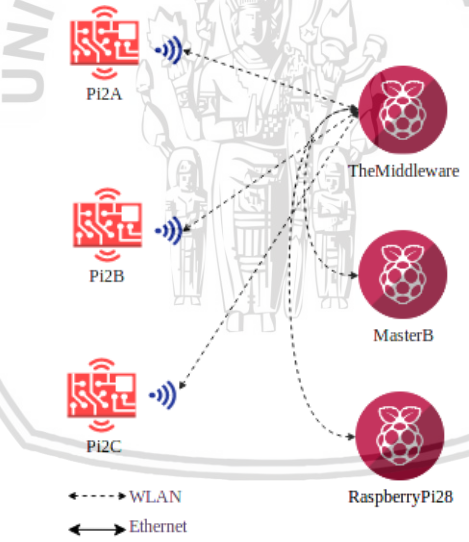
Sistem pada penelitian sebelumnya dapat dilihat pada Gambar 4.2. Lingkungan sistem ini terdiri dari sensor dan *middleware*. Setiap data akan melalui melalui *middleware. Middleware* juga berfungsi sebagai penyedia akses point sehingga trafik seperti SSH, NTP, EAPOL, DNS, MDNS , ARP dan sebagainya akan diperoleh ketika merekam trafik pada jaringan IoT ini.



Gambar 4.2 Lingkungan Penelitian Sebelumnya

4.3.2 Lingkungan Penelitian Penelitian ini

Middleware juga berjalan dapat beberapa perangkat karena pada arsitektur ini *middleware* menggunakan konsep *cluster*. Konsep *cluster* ini akan mengintegrasikan data kepada *middleware* yang terhubung dengan bantuan Redis Cluster. Pada penelitian ini, *middleware* tersebut adalah MasterB, The*Middleware*, dan RaspberryPi28. Penelitian ini menggunakan MasterB sebagai lingkungan implementasi dari IDS.

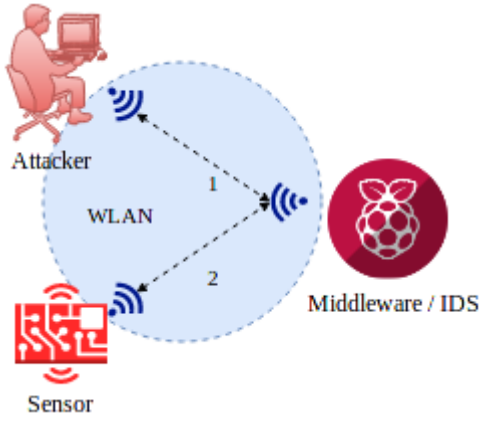


Gambar 4.3 Lingkungan Penelitian Yang Digunakan

4.3.3 Alur Komunikasi Serangan

Alur komunikasi sistem dapat dilihat pada Gambar 4.4. Alur komunikasi serangan ada tiga perangkat utama, yaitu *attacker*, sensor dan *middleware*. Alur komunikasi serangan sendiri dibagi menjadi dua, yaitu :

1. Selama proses pengriman data berlangsung *attacker* akan melakukan serangan DoS pada *middleware*.
2. *Middleware* akan meneritma data dari sensor. Pengiriman data ini dilakukan dengan protokol CoAP dan MQTT.



Gambar 4.4 Alur Serangan

4.4 Kebutuhan *Machine Learning* J48

Machine learning pada penelitian membutuhkan data trafik serangan dan trafik normal. Proses pengambilan trafik akan dilakukan pada *middleware* dengan program tcpdump dan disimpan dengan format pcap. Trafik tersebut akan diambil sesuai dengan daftar berikut ini:

a. Trafik normal dalam waktu 30 menit
b. Trafik serangan DoS TCP ipv4 sebanyak 10000 paket
c. Trafik serangan DoS TCP ipv6 sebanyak 10000 paket
d. Trafik serangan DoS UDP ipv4 sebanyak 10000 paket
e. Trafik serangan DoS UDP ipv6 sebanyak 10000 paket

4.5 Kebutuhan Aplikasi IDS

Bagian ini berisi IDS yang dibutuhkan pada penelitian ini berdasarkan taksonomi yang diperoleh pada sumber literatur. Ada lima parameter yang harus ditentukan jenisnya berdasarkan keuntungan dan kerugiannya. Kelima parameter tersebut adalah lokasi IDS, fungsi IDS, Penempatan IDS, waktu pendeteksi IDS dan mekanisme pendeteksi IDS.

4.5.1 Lokasi IDS

Pada penelitian ini IDS yang digunakan adalah *network-based.* Jenis ini dipilih karena bisa menanggulangi serangan DoS. Selain itu jenis *host-based* dapat dijadikan target dari serangan DoS itu sendiri sehingga kurang handal dalam menanggulangi serangan DoS.

4.5.2 Fungsi IDS

Penelitian ini membutuhkan pendeteksian dan penanggulangan pada jaringan yang akan dilindungi. Selain itu, serangan DoS juga perlu ditanggulangi secara otomatis. IDS yang digunakan juga *network-based* sehingga mendukung implementasi dari IDS.

4.5.3 Penempatan IDS

Penelitian ini menggunakan single *host* karena jaringan yang dilindungi pada lingkungan penelitian hanya terdapat *node* dengan jumlah kecil dan trafik yang masih kecil karena perangkat yang digunakan adalah perangkat IoT.

4.5.4 Waktu Pendeteksian IDS

Jaringan IoT yang dilindungi akan melindungi dari serangan DoS, sehingga mekanisme pendeteksian pada penelitian perlu dilakukan secara *real time* agar dapat ditanggulangi pada saat itu juga.

4.5.5 Mekanisme Pendeteksi IDS

A*nomaly-based* memiliki kelemahan nilai *false positif* yang tinggi namun membutuhkan lebih sedikit *memory*. Jenis ini lebih ringan karena rule pada .

Terdapat beberapa algoritme *decision tree* yang dapat digunakan pada *machine learning*. Pada penelitian sebelumnya, telah dievaluasi penggunaan algoritme pada *machine learning. Decision tree* dipilih karena memiliki kemampuan yang tinggi dalam mendeteksi *anomaly*. Hasil akurasi terbaik untuk mendeteksi serangan dengan *decision tree* ada pada algoritme J48.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan *Machine Learning* J48 Weka

Perancangan dan implementasi menjelaskan tentang bagaimana IDS tersebut dirancang dan diimplementasikan pada arsitektur IoT yang telah ada. Bagian perancangan dan implmentasi tersebut dapat dilihat pada subbab pada bagian ini.

5.1.1 Perancangan Pengambilan Trafik Normal

Data trafik normal didapatkan dengan cara merekam semua paket yang masuk dan keluar dari jaringan yang ingin dilindungi. Kemudian data tersebut akan disimpan ke dalam bentuk json. Adapun trafik yang didapatkan pada saat melakukan survey trafik pada penelitian ini adalah sebagai berikut :

1. SSH menggunakan protokol TCP dengan port 22.
2. Redis menggunakan protokol TCP. Pada penelitian ini terdapat lima *instance* redis dengan port 6379-6383 dan port 16379-16383.
3. MQTT menggunakan protokol TCP dengan port 1883. MQTT pada sistem ini berjalan pada IPv4 dan IPv6.
4. CoAP menggunakan protokol UDP dengan port 5683. CoAP pada sistem ini berjalan pada IPv4 dan IPv6.
5. NTP menggunakan protokol UDP dengan port 123.
6. ICMP menggunakan protokol ICMP untuk IPv4 dan ICMPv6 untuk IPv6.
7. Arp menggunakan protokol ARP.
8. Eapol menggunakan protokol eapol.
9. DNS menggunakan protokol UDP dengan port default 53.
10. MDNS menggunakan protokol UDP dengan port default 5353.
11. IGMP menggunakan protokol IGMPv3.
12. DHCP menggunakan protokol UDP dengan port default 67 dan 68

5.1.2 Perancangan Pengambilan Trafik Serangan

Trafik serangan didapatkan dari memanipulasi serangan terhadap port yang terbuka pada *middleware*. Pada penelitian ini terdapat dua port yang akan diserang yaitu port 1883 dan 5683. Port 1883 adalah port TCP yang digunakan oleh MQTT, sedangkan port 5683 adalah port UDP yang digunakan oleh CoAP. Port TCP akan diserang dengan serangan TCP SYN *flood* sedangkan port UDP akan diserangan dengan UDP *flood*. Skenario penyerangan dapat dilihat pada Tabel 5.1 dan Tabel 5.2.

Tabel 5.1 Skenario Pengambilan Trafik Serangan

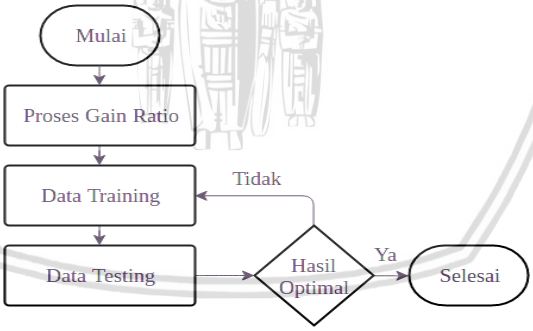
No	Nama	Skenario
1	TCP SYN *Flood* IPv4	Mengirimkan 10000 trafik TCP SYN *Flood* pada port 1883 dengan menggunakan IPv4, panjang data setiap paket ditentukan sebesar 1024 byte, dan port asal acak. Selain itu, korban akan merekam trafik yang diterma ke dalam bentuk pcap

Tabel 5.2 Skenario Pengambilan Trafik Serangan Lanjutan

No	Nama	Skenario
2	TCP SYN *Flood* IPv6	Mengirimkan 10000 trafik TCP SYN *Flood* pada port 1883 dengan menggunakan IPv6, panjang data setiap paket ditentukan sebesar 1024 byte, dan port asal acak. Selain itu, korban akan merekam trafik yang diterma ke dalam bentuk pcap
3	UDP *FLOOD* IPv4	Mengirimkan 10000 trafik UDP *Flood* pada port 5683 dengan menggunakan IPv4, panjang data setiap paket ditentukan sebesar 1024 byte, dan port asal acak. Selain itu, korban akan merekam trafik yang diterma ke dalam bentuk pcap
4	UDP *FLOOD* IPv6	Mengirimkan 10000 trafik UDP *Flood* pada port 5683 dengan menggunakan IPv6, panjang data setiap paket ditentukan sebesar 1024 byte, dan port asal acak. Selain itu, korban akan merekam trafik yang diterma ke dalam bentuk pcap

5.1.3 Perancangan *Decision Tree* Dengan J48

Pada proses *gain ratio*, data yang dikumpulkan pada proses pengambilan data sebelumnya akan dihitung untuk dicari fitur yang paling berpengaruh dalam decision tree. Setelah proses pemilihan fitur selesai, kemudian dilakukan proses data training dengan algoritme J48 untuk dibentuk sebuah decision tree. Decision tree yang terbentuk akan melalui proses data *test*ing untuk mengetahui tingkat akurasinya. Proses ini akan dijalankan berulangkali mulai dari pemilihan fitur sampai data *test*ing untuk mendapatkan akurasi tertinggi. Diagram alir proses ini dapat dilihat pada Gambar 5.1.



Gambar 5.1 Diagram Alir Perancangan Detection Engine

5.1.3.1 Pengumpulan Data

Data yang dikumpulkan dalam bentuk file binary pcap akan dikonversi ke dalam bentuk json agar tidak kehilangan atribut yang dimilki. Selain itu, dengan menggunakan json akan memudahkan proses penggabungan data. Setelah data dalam bentuk json dikumpulkan, kemudian akan dikonversi ke format csv agar dapat dibaca oleh aplikasi weka.

Berdasarkan rancangan pengambilan trafik pada bagian 5.1.1 dan 5.1.2, akan didapatkan lima file yang berbeda. Deskripsi file tersebut dapat dilihat pada Tabel 5.3. File tersebut akan diproses untuk menghasilkan file csv.

Tabel 5.3 File JSON

No	Nama	Deskripsi
1	DOS_TCP4.json	Traffik dari serangan DoS SYN *flood* IPv4
2	DOS_TCP6.json	Traffik dari serangan DoS SYN *flood*
3	DOS_UDP4.json	Traffik dari serangan DoS UDP *flood* IPv4
4	DOS_UDP6.json	Traffik dari serangan DoS UDP *flood* IPv6
5	NORMAL.json	Traffik normal

5.1.3.2 Pemilihan Fitur

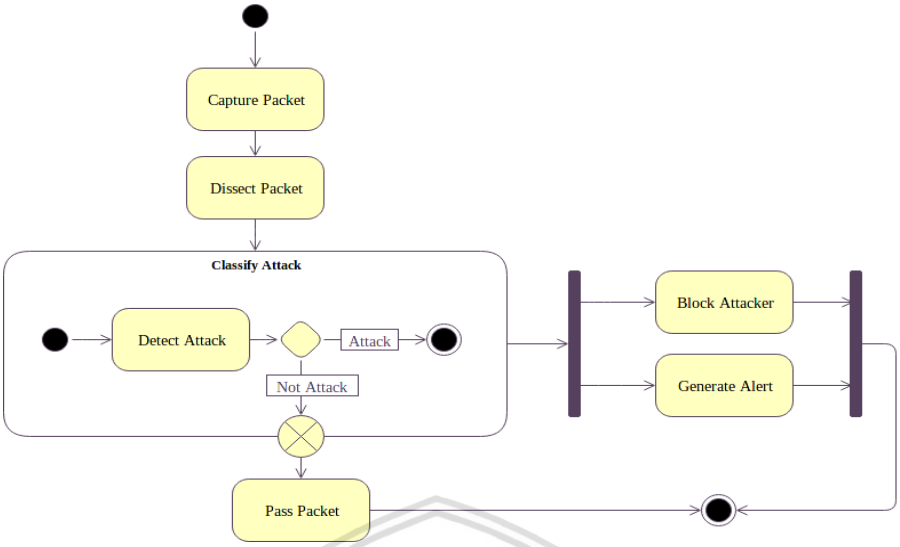
Dalam pengambilan data, terdapat beberapa fitur yang dapat mengganggu atau disebut sebagai *data noise* dan *data redundant*. Fitur ini menganggu proses perhitungan pada *machine learning*. Fitur tersebut akan dimodifikasi atau dihilangkan sehingga mendapatkan hasil optimal dari akurasi *machine learning*. Pada proses ini juga beberapa fitur akan dihapus. Fitur-fitur yang dihapus adalah fitur dengan variasi yang tinggi dan tidak relevan untuk digunakan sebagai parameter pendeteksi serangan. Contoh fitur yang dihapus adalah fitur TCP *checksum* dan fitur MAC *address*. Data yang dikumpulkan akan diproses dengan aplikasi weka. Kemudian fitur terbaik akan dirangking dengan menggunakan algoritme IGR. Setelah hasil IGR didapatkan, akan dilakukan proses pembuatan *decision tree*.

5.1.3.3 Decision Tree Dengan Algoritme J48

Hasil dari proses pemilihan fitur akan dilanjutkan dengan proses *data training* dan *data testing* dengan algoritme J48. Proses ini akan dilakukan berulang-ulang dengan mengubah jumlah fitur yang digunakan untuk mencari *decision tree* dengan akurasi terbaik. Hasil tersebut akan diimplementasikan dengan menggunakan bahasa python. File python ini akan digunakan sebagai detection engine dari IDS pada penelitian ini.

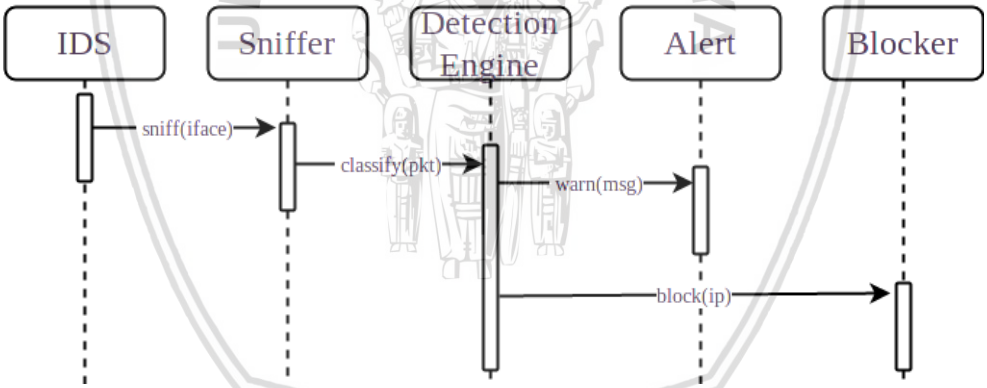
5.2 Perancangan Aplikasi IDS

Aplikasi IDS meliputi proses *capture packet, dissect pakcet, detect atack, generate alert* dan *block attacker*. State *machine* diagram aplikasi IDS ini dapat dilihat pada Gambar 5.2.



Gambar 5.2 State *Machine* IDS

Aplikasi IDS memerlukan module *sniffer*, *detection engine*, *alert* dan *blocker*. IDS akan melakukan pengambilan paket pada sebuah interface dengan module sniffer. Setiap paket yang masuk kepada sniffer akan diproses oleh detection engine. Apabila *detection engine* mendeteksi serangan, maka akan memanggil *module alert* dan *blocker*. *Sequence diagram* proses ini dapat dilihat pada Gambar 5.3.



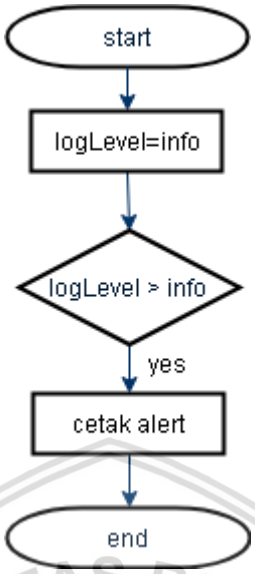
Gambar 5.3 *Sequence Diagram* IDS

5.2.1 Perancangan *Detection Engine* IDS

Salah satu komponen utama dari IDS adalah *detection engine* yang dipakai. *Detection Engine* adalah bagian yang akan menentukan keberhasilan dari sebuah IDS dalam mendeteksi serangan. Pembentukan *detection engine* pada penelitian ini menggunakan *machine learning.*

Machine learning dibangun dengan melalui proses pengumpulan data, pemilihan fitur, pembentukan *decision tree*, *data training* dan *data testing*. Dari proses ini akan dihasilkan *decision tree* dengan akurasi terbaik. *Decision tree* terbaik tersebut akan digunakan sebagai *detection engine* pada IDS penilitian ini.

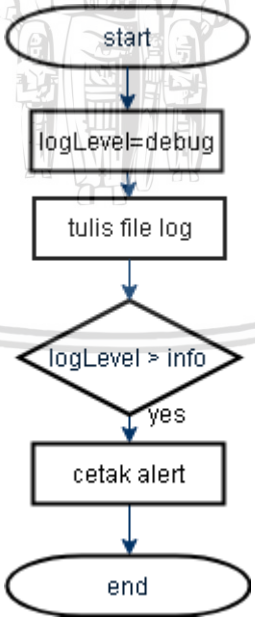
5.2.2 Perancangan *Alert* IDS



Gambar 5.4 Flowchart perancangan Albert IDS

Pada gambar 5.4 menunjukkan bahwa pada kondisi log level sama dengan info maka *alert* tidak akan tampil. Namun apabila log level lebih besar dari info, maka *alert* akan tampil.

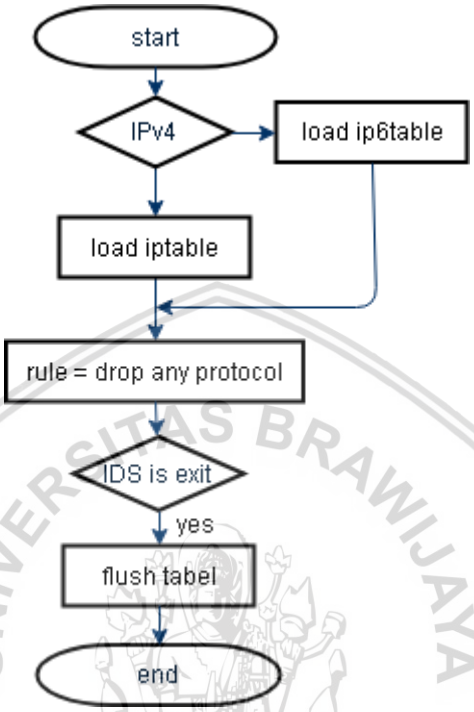
5.2.3 Perancangan Logger IDS



Gambar 5.5 Flowchart perancangan logger IDS

Pada gambar 5.5 menunjukkan kondisi ketika log level sama dengan debug, tahap selanjutnya yaitu penulisan file log. Ketika log level lebih besar dari info maka akan menampilkan *alert*.

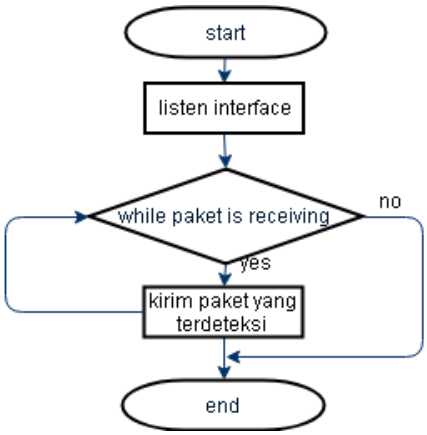
5.2.4 Perancangan Blocker IDS



Gambar 5.6 Flowchart perancangan blocker IDS

Pada gambar 5.6 menunjukkan diagram alir yang terdapat pada kondisi penggunaan IPv4 melakukan *load iptable* sedangkan untuk IPv6 melakukan *load* ip6table. Tahap selanjutnya yaitu penentuan rute, di mana beberapa protokol akan didrop. Selanjutnya ada kondisi jilat IDS telah keluar, maka flush table akan dilakukan.

5.2.5 Perancangan *Network* Sniffer IDS



Gambar 5.7 Flowchart perancangan Networks sniffer IDS

Gambar 5.7 menunjukkan proses *listen interface* yang selanjutnya memasuki kondisi di mana selama paket masih diterima, maka kiriman paket akan terus terdeteksi. Namun ketika paket telah berhenti diterima , maka proses sniffer akan berakhir.

5.3 Perancangan Pengujian Kinerja IDS

Pengujian kinerja adalah pengujian untuk mencari tahu pengaruh aplikasi IDS pada perangkat *middleware*. Pengujian ini juga melibatkan kecepatan IDS dalam mengklasifikasikan paket.

5.3.1 Perancangan Pengujian Penggunaan CPU Pada *Middleware*

Pengujian ini dilakukan dengan cara merekam CPU pada *middleware* ketika trafik normal dan ketika terjadi serangan. Pengujian ini dapat dilihat pada Tabel 5.4.

Tabel 5.4 Pengujian Penggunaan CPU Pada *Middleware*

Kode	Pengujian	Skenario
KC_01	Penggunaan CPU pada Trafik Normal	1. Menjalankan IDS pada *middleware* 2. Merekam CPU pada trafik normal selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Melakukan perhitungan CPU
KC_02	Penggunaan CPU pada Serangan UDP *flood* IPv4	1. Menjalankan IDS pada *middleware* 2. Merekam CPU selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan UDP *flood* IPv4 5. Melakukan perhitungan CPU
KC_03	Penggunaan CPU pada Serangan UDP *flood* IPv6	1. Menjalankan IDS pada *middleware* 2. Merekam CPU selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit

		4. Mengirimkan trafik serangan UDP *flood* IPv6 5. Melakukan perhitungan CPU
KC_04	Penggunaan CPU pada Serangan SYN *flood* IPv4	1. Menjalankan IDS pada *middleware* 2. Merekam CPU selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan SYN *flood* IPv4 5. Melakukan perhitungan CPU
KC_05	Penggunaan CPU pada Serangan SYN *flood* IPv6	1. Menjalankan IDS pada *middleware* 2. Merekam CPU selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan SYN *flood* IPv6 5. Melakukan perhitungan CPU

5.3.1 Perancangan Pengujian Penggunaan *Memory* Pada *Middleware*

Pengujian ini dilakukan dengan cara merekam *memory* pada *middleware* ketika trafik normal dan ketika terjadi serangan. Pengujian ini dapat dilihat pada Tabel 5.5.

Tabel 5.5 Pengujian Penggunaan *Memory* Pada *Middleware*

Kode	Pengujian	Skenario
KM_01	Penggunaan *memory* pada Trafik Normal	1. Menjalankan IDS pada *middleware* 2. Merekam *memory* pada trafik normal selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Melakukan perhitungan *memory*
KM_02	Penggunaan *memory* pada Serangan UDP *flood* IPv4	1. Menjalankan IDS pada *middleware* 2. Merekam *memory* selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan UDP *flood* IPv4 5. Melakukan perhitungan *memory*
KM_03	Penggunaan *memory* pada Serangan UDP *flood* IPv6	1. Menjalankan IDS pada *middleware* 2. Merekam *memory* selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan UDP *flood* IPv6 5. Melakukan perhitungan *memory*
KM_04	Penggunaan *memory* pada Serangan SYN *flood* IPv4	1. Menjalankan IDS pada *middleware* 2. Merekam *memory* selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan SYN *flood* IPv4 5. Melakukan perhitungan *memory*
KM_05	Penggunaan *memory* pada Serangan SYN *flood* IPv6	1. Menjalankan IDS pada *middleware* 2. Merekam *memory* selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan SYN *flood* IPv6 5. Melakukan perhitungan *memory*

5.3.2 Perancangan Pengujian Kecepatan Klasifikasi Paket

Pengujian ini dilakukan dengan menghitung waktu klasifikasi paket pada *middleware* ketika trafik normal dan ketika terjadi serangan. Pengujian ini dapat dilihat pada Tabel 5.6.

Tabel 5.6 Pengujian Kecepatan Klasifikasi Paket

Kode	Pengujian	Skenario
KK_01	Kecepatan klasifikasi pada Trafik Normal	1. Menjalankan IDS pada *middleware* 2. Menghitung waktu proses paket pada trafik normal selama 10 menit pada *middleware* 3. Melakukan perhitungan rata-rata kecepatan klasifikasi paket
KK _02	Kecepatan klasifikasi pada Serangan UDP *flood* IPv4	1. Menjalankan IDS pada *middleware* 2. Menghitung waktu proses paket pada trafik serangan UDP *flood* IPv4 selama 10 menit pada *middleware* 3. Melakukan perhitungan rata-rata kecepatan klasifikasi paket
KK _03	Kecepatan klasifikasi pada Serangan UDP *flood* IPv6	1. Menjalankan IDS pada *middleware* 2. Menghitung waktu proses paket pada trafik serangan UDP *flood* IPv6 selama 10 menit pada *middleware* 3. Melakukan perhitungan rata-rata kecepatan klasifikasi paket
KK _04	Kecepatan klasifikasi pada Serangan SYN *flood* IPv4	1. Menjalankan IDS pada *middleware* 2. Menghitung waktu proses paket pada trafik serangan SYN *flood* IPv4 selama 10 menit pada *middleware* 3. Melakukan perhitungan rata-rata kecepatan klasifikasi paket
KK _05	Kecepatan klasifikasi pada Serangan SYN *flood* IPv6	1. Menjalankan IDS pada *middleware* 2. Menghitung waktu proses paket pada trafik serangan SYN *flood* IPv6 selama 10 menit pada *middleware* 3. Melakukan perhitungan rata-rata kecepatan klasifikasi paket

5.4 Perancangan Pengujian Fungsional IDS

Pengujian sistem digunakan untuk menguji aplikasi dengan kesesuaian dengan kebutuhan fungsionalitas. Pengujian ini meliputi pengujian pengujian pengambilan trafik, pengujian akurasi detection engine dan pengujian penanggulangan serangan. Pada tiap fitur tersebut akan dilakukan pengujian *black box*. Pengujian *black box* akan dilakukan dengan menganalisis kesesuain dari *input* dan *output* pada tiap fungsionalitas.

5.4.1 Pengujian Akurasi Pengambilan Trafik IDS

Pengujian akurasi *network sniffer* dilakukan untuk menguji akurasi dari jumlah paket yang dapat ditangkap oleh IDS. Pada penelitian ini IDS yang digunakan untuk merekam paket pada jaringan menggunakan *library* scapy. *Library* ini akan

dibandingkan dengan tcpdump untuk menguji tingkat akurasinya. Pengujian dilakukan setiap lima menit dengan parameter trafik yang berbeda. Kemudian akan dihitung akurasinya dengan Persamaan 5.1.

$$Akurasi\ Trafik\ IDS = \frac{\sum scapy}{\sum tcpdump}\ x\ 100\% \quad (5.1)$$

Pengujian dilakukan sebanyak lima kali. Pengujian ini dilakukan berdasarkan parameter dari trafik yang akan diuji. Adapaun parameter tersebut adalah trafik normal, trafik serangan UDP *flood* IPv4, trafik serangan UDP *flood* IPv6, trafik serangan SYN *flood* IPv4 dan trafik serangan SYN *flood* IPv6. Tabel 5.7 menjelaskan tentang skenario pengujian pada bagian ini.

Tabel 5.7 Pengujian Akurasi Pengambilan Trafik IDS

Kode	Pengujian	Skenario
AT_01	Akurasi pengambilan trafik pada Trafik Normal	1. IDS diset untuk menampilkan pesan setiap ada paket yang masuk 2. Sensor mengirimkan paket setiap satu menit 3. IDS dan tcpdump dijalankan di *middleware* secara bersamaan selama lima menit 4. Jumlah pesan paket masuk pada IDS dihitung 5. Menghitung paket masuk pada IDS 6. Melakukan komparasi antara hasil dari IDS dengan tcpdump
AT_02	Akurasi pengambilan trafik pada Serangan UDP *flood* IPv4	1. IDS diset untuk menampilkan pesan setiap ada paket yang masuk 2. Sensor mengirimkan paket setiap satu menit 3. IDS dan tcpdump dijalankan di *middleware* secara bersamaan selama lima menit 4. *Middleware* diserang denga UDP *flood* IPv4 5. Jumlah pesan paket masuk pada IDS dihitung 6. Menghitung paket masuk pada IDS 7. Melakukan komparasi antara hasil dari IDS dengan tcpdump
AT_03	Akurasi pengambilan trafik pada Serangan UDP *flood* IPv6	1. IDS diset untuk menampilkan pesan setiap ada paket yang masuk 2. Sensor mengirimkan paket setiap satu menit 3. IDS dan tcpdump dijalankan di *middleware* secara bersamaan selama lima menit 4. *Middleware* diserang denga UDP *flood* IPv6 5. Jumlah pesan paket masuk pada IDS dihitung 6. Menghitung paket masuk pada IDS 7. Melakukan komparasi antara hasil dari IDS dengan tcpdump
AT_04	Akurasi pengambilan trafik pada Serangan SYN *flood* IPv4	1. IDS diset untuk menampilkan pesan setiap ada paket yang masuk 2. Sensor mengirimkan paket setiap satu menit 3. IDS dan tcpdump dijalankan di *middleware* secara bersamaan selama lima menit 4. *Middleware* diserang denga SYN *flood* IPv4 5. Jumlah pesan paket masuk pada IDS dihitung

		6. Menghitung paket masuk pada IDS 7. Melakukan komparasi antara hasil dari IDS dengan tcpdump
AT_05	Akurasi pengambilan trafik pada Serangan SYN *flood* IPv6	1. IDS diset untuk menampilkan pesan setiap ada paket yang masuk 2. Sensor mengirimkan paket setiap satu menit 3. IDS dan tcpdump dijalankan di *middleware* secara bersamaan selama lima menit 4. *Middleware* diserang denga SYN *flood* IPv6 5. Jumlah pesan paket masuk pada IDS dihitung 6. Menghitung paket masuk pada IDS 7. Melakukan komparasi antara hasil dari IDS dengan tcpdump

5.4.2 Pengujian Akurasi *Detection Engine* IDS

Pengujian akurasi *detection engine* digunakan untuk menguji jumlah paket yang dapat diklasifikasikan oleh IDS. Pengujian akurasi *detection engine* dilakukan dengan menggunakan rumus pada Persamaan 5.2 dengan deskripsi TP adalah *true positive*, FN adalah *false negative*, TN adalah *true negative* dan FN adalah *false negative.*

$$Akurasi\ Detection\ Engine = \frac{\sum TP+\sum FN}{\sum TP+\sum TN+\sum FP+\sum FN} \times 100\% \quad (5.2)$$

Tabel 5.8 menjelaskan tentang skenario pengujian pada bagian ini. Pengujian ini dilakukan ketika trafik normal dan ketika terjadi serangan.

Tabel 5.8 Pengujian Akurasi *Detection Engine* IDS

Kode	Pengujian	Skenario
DE_01	Akurasi *detection engine* pada Trafik Normal	1. IDS dijalankan dalam waktu 30 menit 2. Sensor mengirimkan data dalam waktu satu menit 3. Melihat summary dari paket yang diklasifikasikan oleh IDS
DE_02	Akurasi detection engine trafik pada Serangan UDP *flood* IPv4	1. IDS dijalankan dalam waktu 30 menit 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv4 paket 3. IDS dimatikan dan diihat summary dari paket yang diklasifikasikan oleh IDS
DE_03	Akurasi detection engine trafik pada Serangan UDP *flood* IPv6	1. IDS dijalankan dalam waktu 30 menit 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv6 paket 3. IDS dimatikan dan diihat summary dari paket yang diklasifikasikan oleh IDS
DE_04	Akurasi detection engine trafik pada Serangan SYN *flood* IPv4	1. IDS dijalankan dalam waktu 30 menit 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 TCP IPv4 paket 3. IDS dimatikan dan diihat summary dari paket yang diklasifikasikan oleh IDS

DE_05	Akurasi detection engine trafik pada Serangan SYN *flood* IPv6	1. IDS dijalankan dalam waktu 30 menit 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 TCP IPv6 paket 3. IDS dimatikan dan diihat summary dari paket yang diklasifikasikan oleh IDS

5.4.3 Pengujian Penanggulangan Serangan IDS

Pengujian penanggulangan serangan digunakan untuk menguji penanggulangan serangan pada IDS. Pada Pengujian ini dilakukan pada saat trafik serangan. Tabel 5.9 menjelaskan tentang skenario pengujian pada bagian ini.

Tabel 5.9 Pengujian Penanggulangan Serangan IDS

Kode	Pengujian	Skenario
PS_01	Kemampuan menanggulangi terhadap Serangan UDP *flood* IPv4	1. IDS dijalankan 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv4 paket 3. Melakukan cek *iptables* pada sistem operasi
PS_02	Kemampuan menanggulangi terhadap Serangan UDP *flood* IPv6	1. IDS dijalankan 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv6 paket 3. Melakukan cek *iptables* pada sistem operasi
PS_03	Kemampuan menanggulangi terhadap Serangan SYN *flood* IPv4	1. IDS dijalankan 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 SYN IPv4 paket 3. Melakukan cek *iptables* pada sistem operasi
PS_04	Kemampuan menanggulangi terhadap Serangan SYN *flood* IPv6	1. IDS dijalankan 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 SYN IPv6 paket 3. Melakukan cek *iptables* pada sistem operasi

5.4.4 Pengujian Akurasi *Alert* IDS

Pengujian akurasi *detection engine* digunakan untuk menguji jumlah paket yang dapat diklasifikasikan oleh IDS. Pengujian akurasi *detection engine* dilakukan dengan menggunakan rumus pada Persamaan 5.3 dengan deskripsi TP adalah *true positive*, FN adalah *false negative*, TN adalah *true negative* dan FN adalah *false negative.*

$$Akurasi\ Alert = \frac{\sum TP+\sum FN}{\sum TP+\sum TN+\sum FP+\sum FN} \times 100\% \quad (5.3)$$

Tabel 5.10 menjelaskan tentang skenario pengujian pada bagian ini. Pengujian ini dilakukan ketika trafik normal dan ketika terjadi serangan.

Tabel 5.10 Pengujian Akurasi *Alert* IDS

Kode	Pengujian	Skenario
AA_01	Kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv4	1. IDS dijalankan dan output aplikasi disimpan dalam sebuah file 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv4 paket 3. Melakukan melakukan perhitungan jumlah *alert* yang diberikan dari file tersebut.
AA_02	Kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv6	1. IDS dijalankan dan output aplikasi disimpan dalam sebuah file 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv6 paket 3. Melakukan melakukan perhitungan jumlah *alert* yang diberikan dari file tersebut.
AA_03	Kemampuan menampilkan *alert* terhadap Serangan SYN *flood* IPv4	1. IDS dijalankan dan output aplikasi disimpan dalam sebuah file 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 SYN IPv4 paket 3. Melakukan melakukan perhitungan jumlah *alert* yang diberikan dari file tersebut.
AA_04	Kemampuan menampilkan *alert* terhadap Serangan SYN *flood* IPv6	1. IDS dijalankan dan output aplikasi disimpan dalam sebuah file 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 SYN IPv6 paket 3. Melakukan melakukan perhitungan jumlah *alert* yang diberikan dari file tersebut.

5.4.5 Pengujian *Logging* IDS

Pengujian *logging* digunakan untuk menguji *logging* pada IDS. Pada Pengujian ini dilakukan dengan mengecek file log yang dibuat oleh ids. Tabel 5.11 menjelaskan tentang skenario pengujian pada bagian ini.

Tabel 5.11 Pengujian *Logging* IDS

Kode	Pengujian	Skenario
LG_01	Pengujian Logging	1. Melakukan pengecekan terhadap log file selama menjalankan aplikasi IDS.

5.5 Perancangan Pengujian Non-Fungsional

Pengujian non-fungsional dapat digunakan untuk memabtu meningkatkan keberhasilan dari implementasi. Pengujian ini dibagi menjadi dua yaitu *availability* dan *performance.*

5.5.1 Pengujian *Availability* IDS

Pengujian ini akan digunakan untuk menguji kemampuan IDS untuk tetap dapat berjalan dalam waktu yang lama. Pada Tabel 5.12 menunjukkan scenario dari pengujian pada bagian ini.

Tabel 5.12 Pengujian *Availability* IDS

Kode	Pengujian	Skenario
AV_01	Sistem Dapat Berjalan Lebih Dari 1 Jam	1. Menjalankan IDS sebagai sebuah service 2. Mengecek status IDS keesokan harinya.
AV_02	Auto Restart	1. Menjalankan IDS sebagai sebuah service 2. Mematikan paksa IDS 3. Mengecek status IDS

5.5.2 Pengujian *Performance* IDS

Pengujian ini akan digunakan untuk menguji kecepatan IDS dalam mengklasifikaiskan paket. Pada Tabel 5.13 menunjukkan scenario dari pengujian pada bagian ini.

Tabel 5.13 Pengujian *Performance* IDS

Kode	Pengujian	Skenario
PF_01	Mengklasifikasi paket kurang dari satu detik	1. Menyimpan waktu ketika paket masuk dalam *detection engine* dan waktu ketika selesai diklasifikasi 2. Menghitung waktu rata-rata yang dibutuhkan dalam 5000 paket yang diterima

5.6 Implementasi *Machine Learning*

Proses pengambilan data akan menghasilkan dua data. Yaitu merekam trafik pada jaringan dan merekam sumber daya yang dipakai pada sistem ini. Trafik yang direkam sesuai dengan skenario pada bagian perancangan pengambilan trafik.

5.6.1 Implementasi Pengambilan Trafik Normal

Pengambilan data trafik normal dilakukan dengan merekam semua trafik pada jaringan selama 30 menit dengan aplikasi tcpdump. Gambar 5.4 menunjukkan cara mengambil trafik pada jaringan.

```
pi@masterB:~/hilman/traffic $ sudo timeout 1800 tcpdump -i wlan0 -w normal.pcap
```

Gambar 5.8 Pengambilan Data Trafik Normal *Middleware*

Gambar 5.5 menunjukkan crontab dari sensor ketika mengirim data. Crontab akan mengeksekusi *script* setiap satu menit. Crontab akan mengirimkan paket MQTT dan CoAP secara sekuensial.

```
*/1 * * * * python /home/pi/hilman/sentMasterB/mqtt/sentMqtt.py  &&  sudo node /
home/pi/hilman/sentMasterB/concurrent/concurrentCoAP.js
```

Gambar 5.9 Crontab Pengiriman Data Sensor

5.6.2 Implementasi Pengambilan Trafik Serangan

Trafik serangan dihasilkan dengan *flood*ing TCP SYN atau UDP. Trafik tersebut dibuat dengan *script* python. *Pseudocode* dari *script* yang akan digunakan untuk *flood*ing dapat dilihat pada Tabel 5.14.

Tabel 5.14 *Pseudocode*DoS

Psuedocode

type = argument.type, ip = argument.ip, port = argument.port, count = argument.count
iface = argument.iface, worker = core.count()
if ipaddress.version == IPv4 then
  mac = send(arp(dst=ipaddress.str())).getMacResponse()
else if ipaddress.version == IPv6 then
  mac = getMacAddressFromIPv6(ipaddess.str())
if port != null then
  if type == “udp” then
  transportheader = UDP(dport=port, sport=Rand())
  else if type == “tcp” then
  transportheader = TCP(dport=port, flags="S", sport=Rand())
else then
  return “no destination port given”
packet = Ether(dst=mac)| IP(dst= ipaddress.str())| transportheader | Raw(Rand())
if count % worker not 0 then
  extraworker = true
socket = create*Layer*2Socket(iface)
for eachworker in multiprocess(worker) then
  new thread(socket.send(packet, count=count/worker, iface=iface))
If extraworker then
  new thread(socket.send(packet, count=count%worker, iface=iface))

5.6.2.1 SYN *Flood* IPv4

Pada proses ini membutuhkan dua buah *node*. *Node* pertama sebagai *middleware* dan *node* kedua sebagai penyerang. Gambar 5.6 menunjukkan proses pengiriman paket dari penyerang. Paket akan dikirimkan selama lima menit.

```
pi@raspberrypi43:~ $ sudo timeout 300 python3 gflood.py -i wlan0 -dhost 192.168.
42.100 -tcp -dport 1883 -c 100000 -send 3
```

Gambar 5.10 Pengiriman SYN *Flood* IPv4

Implementasi dari *library* python untuk mengirimkan pesan SYN akan membuat kernel mengirimkan pesan RST untuk menutup koneksi. Untuk mencegah pesan RST dikirimkan dapat menggunakan *firewall* untuk melakukan drop packet yang dapat dilihat pada Gambar 5.7.

```
pi@raspberrypi43:~ $ sudo iptables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP
```

Gambar 5.11 Block Pesan RST IPv4

5.6.2.2 SYN *Flood* IPv6

Pada proses ini membutuhkan dua buah *node*. *Node* pertama sebagai *middleware* dan *node* kedua sebagai penyerang. Gambar 5.8 menunjukkan proses pengiriman paket dari penyerang. Paket akan dikirimkan selama lima menit.

```
pi@raspberrypi43:~ $ sudo timeout 300 python3 gflood.py -i wlan0 -dhost fe80::ba
27:ebff:feac:1d8b -tcp -dport 1883 -c 100000 -send 3
```

Gambar 5.12 Pengiriman SYN *Flood* IPv6

Implementasi dari library python untuk mengirimkan pesan SYN akan membuat kernel mengirimkan pesan RST untuk menutup koneksi. Untuk mencegah pesan RST dikirimkan dapat melihat Gambar 5.9.

```
pi@raspberrypi43:~ $ sudo ip6tables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP
packet_write_wait:
```

Gambar 5.13 Block Pesan RST IPv6

5.6.2.3 UDP *Flood* IPv4

Pada proses ini membutuhkan dua buah *node*. *Node* pertama sebagai *middleware* dan *node* kedua sebagai penyerang. Gambar 5.10 menunjukkan proses pengiriman paket dari penyerang. Paket akan dikirimkan selama lima menit.

```
pi@raspberrypi43:~ $ sudo timeout 300 python3 gflood.py -i wlan0 -dhost 192.168.
42.100 -udp -dport 5683 -c 100000 -send 3
```

Gambar 5.14 Pengiriman UDP *Flood* IPv4

5.6.2.4 UDP *Flood* IPv6

Pada proses ini membutuhkan dua buah *node*. *Node* pertama sebagai *middleware* dan *node* kedua sebagai penyerang. Gambar 5.11 menunjukkan proses pengiriman paket dari penyerang. Paket akan dikirimkan selama lima menit.

```
pi@raspberrypi43:~ $ sudo timeout 300 python3 gflood.py -i wlan0 -dhost fe80::ba
27:ebff:feac:1d8b -udp -dport 5683 -c 100000 -send 3
```

Gambar 5.15 Pengiriman UDP *Flood* IPv6

5.6.3 Implementasi Decision Tree Dengan Weka J48

Detection engine dibuat melalui proses pengumpulan data, konversi data, pemilihan fitur, *data training* dan *data testing*. Proses perancangan pengumpulan data telah dilakukan pada Bab 5.6.1 dan Bab 5.6.2. Proses konversi data, pemilihan fitur, *data training* dan *data testing* akan dijelaskan pada sub bab pada bagian ini.

5.6.3.1 Implementasi Pengumpulan Data

Trafik yang telah direkam dalam format pcap akan dikonversi ke dalam format JSON dan kemudian disatukan ke format CSV. *Pseudo code* yang digunakan untuk melakukan konversi data dapat dilihat pada Tabel 5.15.

Tabel 5.15 *Pseudocode*Konversi Data

Psuedocode

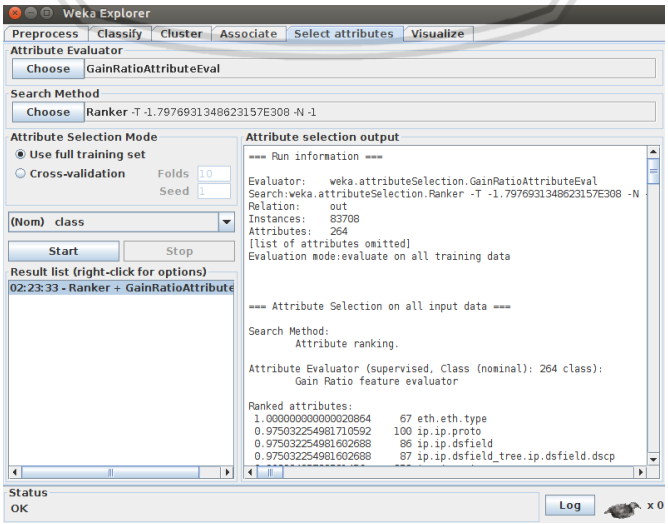
data = []
files = getFilesFromDir('.json')
for file in files then
  json = parsejson(open(file))
  json['class'] = file.filename.del('.json')
  for jsonitem in json then
  jsonitem.pop('frame')
  data.append(jsonitem)
dataframe = normalize(data)
dataframe.del('\'', '\"')
dataframe.setLastColumn('class')
dataframe.tocsv('out')

Gambar 5.12 menampilkan implementasi konversi data yang dilakukan dengan merujuk pada Tabel 5.15. Hasil yang didapatkan pada proses ini adalah dataset yang telah diklasifikasikan dan dikumpulkan menjadi satu dalam format CSV.

Gambar 5.16 Proses Konversi Data

5.6.3.2 Implementasi Pemilihan Fitur Dengan Gain Ratio

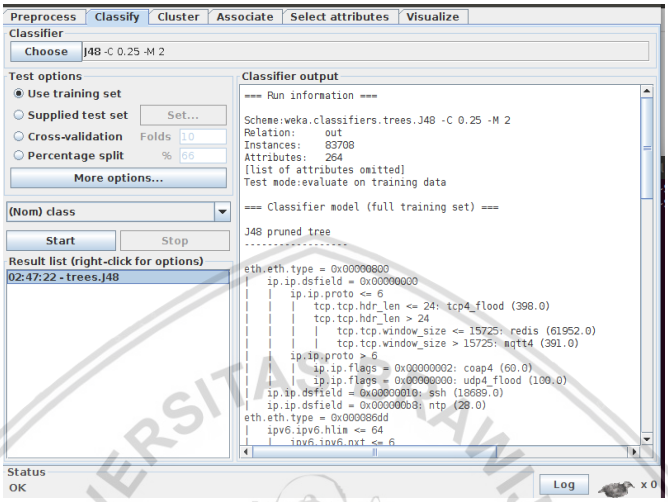
Pemilihan fitur adalah proses penyeleksian fitur yang digunakan untuk menghilangkan *data noise* atau *data redundant* yang ada pada *dataset*. Data akan dilakukan pembersihan dengan menghilangkan data yang memiliki variasi tinggi agar hasil yang didapatkan pada proses *data training* menjadi optimal. Kemudian akan dipilih fitur pada data berdasarkan pengaruhnya dengan menggunakan algoritme IGR. Gambar 5.13 menunjukkan aplikasi weka yang digunakan untuk melakukan proses pemilihan fitur dengan IGR.



Gambar 5.17 Pemilihan Fitur Dengan IGR

5.6.3.3 Implementasi Decision Tree Dengan J48

Pada bagian ini akan dilakukan proses data training untuk membuat model decision tree dengan algoritme J48. Proses ini akan dilakukan berulang kali dengan jumlah fiur yang berbeda berdasarkan rangking dari IGR. Hasil terbaik akan digunakan untuk detection engine pada IDS ini. Adapun gambar dari Weka untuk proses ini dapat dilhat pada Gambar 5.14.



Gambar 5.18 Data Training dan *Testing* Weka

Setelah hasil akurasi dari setiap jumlah fitur didapatkan, data akurasi akan dikumpulkan dan akan dipilih nilai terbaik dari setiap proses yang dilakukan sebelumnya, Tabel 5.16 menunjukkan akurasi yang didapatkan dari proses data training sebelumnya.

Tabel 5.16 Hasil Akurasi yang Didapatkan

No	Skenario	Akurasi
1	Akurasi detection engine dengan 1 fitur	79.3269%
2	Akurasi detection engine dengan 2 fitur	91.9182%
3	Akurasi detection engine dengan 3 fitur	91.3680%
4	Akurasi detection engine dengan 5 fitur	99.9116%
5	Akurasi detection engine dengan 10 fitur	99.9116%
6	Akurasi detection engine dengan 15 fitur	99.9269%
7	***Akurasi detection engine dengan 20 fitur***	**100%**
8	Akurasi detection engine dengan semua fitur	100%

Akurasi terbaik didapatkan dengan menggunakan 20 fitur. Adapun *decision tree* terbaik yang didapatkan pada proses tersebut dapat dilihat pada Tabel 5.17. *Decision tree* terbaik yang didapatkan akan digunakan sebagai rule pada IDS. *Decision tree* ini didapatkan dari hasil proses pada aplikasi Weka yang dilakukan sebelumnya.

Tabel 5.17 Decision Tree dari 20 Fitur

Output Weka
eth.eth.type = 0x000086dd | ipv6.ipv6.plen <= 1032

```
|   |   ipv6.ipv6.plen <= 325: normal (197.0)
|   |   ipv6.ipv6.plen > 325: udp6_flood (10000.0)
|   ipv6.ipv6.plen > 1032: tcp6_flood (10000.0)
eth.eth.type = 0x00000800
|   ip.ip.flags = 0x00000002: normal (77307.0)
|   ip.ip.flags = 0x00000000
|   |   ip.ip.len <= 1052
|   |   |   ip.ip.len <= 689: normal (67.0)
|   |   |   ip.ip.len > 689: udp4_flood (10000.0)
|   |   ip.ip.len > 1052: tcp4_flood (10000.0)
eth.eth.type = 0x00000806: normal (44.0)
eth.eth.type = 0x0000888e: normal (6.0)
```

5.7 Implementasi Aplikasi IDS

Implementasi IDS terdiri dari implementasi *detection engine, alert, logger, blocker* dan *network sniffer.*

5.7.1 Implementasi *Detection Engine*

Pseudocode implementasi dari decision tree terbaik yang didapatkan dapat dilihat pada Tabel 5.18. *Pseudocode*ini adalah *detection engine* yang akan digunakan pada IDS pada penelitin ini. *Detection engine* ini akan melakukan pengecekan pada setiap *header* pada paket yang masuk kemudian melakukan seleksi berdasarkan *decision tree* yang telah dibuat sehingga dapat mendeteksi serangan yang masuk.

Tabel 5.18 *Pseudocode* Decision Tree

Psudocode Decision Tree

```
if pkt[Ether].type==int(b'0x800',16):
   ip = pkt[IP].src
elif pkt[Ether].type==int(b'0x86DD',16):
   ip = pkt[IPv6].src
if getattr(pkt[Ether], 'type') == int(b'0x000086dd',16):
   if getattr(pkt[IPv6], 'plen') <= 1032:
      if getattr(pkt[IPv6], 'plen') <= 325:
         detected = +PACKET_LIST['NORMAL']
      elif getattr(pkt[IPv6], 'plen') > 325:
         detected = +PACKET_LIST['MALICIOUS']['UDP6']
         logger.warning(detected)
   elif getattr(pkt[IPv6], 'plen') > 1032:
      detected = +PACKET_LIST['MALICIOUS']['TCP6']
      logger.warning(detected)
elif getattr(pkt[Ether], 'type') == int(b'0x00000800',16):
   if getattr(pkt[IP], 'flags') == int('0x00000002',16):
      detected = +PACKET_LIST['NORMAL']
   elif getattr(pkt[IP], 'flags') == int('0x00000000',16):
      if getattr(pkt[IP], 'len') <= 1052:
         if getattr(pkt[IP], 'len') <= 689:
            detected = +PACKET_LIST['NORMAL']
         elif getattr(pkt[IP], 'len') >= 689:
            detected = +PACKET_LIST['MALICIOUS']['UDP4']
            logger.warning(detected)
```

```
        elif getattr(pkt[IP], 'len') > 1052:
            detected = +PACKET_LIST['MALICIOUS']['TCP4']
            logger.warning(detected)
elif getattr(pkt[Ether], 'type') == int(b'0x00000806',16):
    detected = +PACKET_LIST['NORMAL']
elif getattr(pkt[Ether], 'type') == int(b'0x0000888e',16):
    detected = +PACKET_LIST['NORMAL']
if detected != PACKET_LIST['NORMAL']:
    if detected == PACKET_LIST['MALICIOUS']['UNDETECTED']:
        +PACKET_LIST['MALICIOUS']['UNDETECTED']
        logger.warning(detected)
    else:
        block(ip)
return PACKET_LIST
```

5.7.2 Implementasi *Alert*

Alert akan menampilkan pesan *alert* ke console dengan *level logging* diatas *info*, serta akan menampilkan informasi tambahan ketika file tersebut dipanggil. *Psudo code* proses ini dapat dilihat pada Tabel 5.19.

Tabel 5.19 *Pseudocode Alert*

Psudocode *Alert* dan Logger

Console.loglevel(info)
if log.level greater than info
  trace.linenumber()
  print time() + *alert*.msg()
print time() + *alert*.msg()

5.7.3 Implementasi Logger

Logging akan menulis aktifitas ke dalam sebuah file teks dengan *level debug*. *Psudo code* proses ini dapat dilihat pada Tabel 5.20.

Tabel 5.20 *Pseudocode Logger*

Psudocode *Alert* dan Logger

file.loglevel(debug)
file.writeTo('/var/log/app-raspyids.log')
if log.level greater than info
  trace.linenumber()
  print time() + *alert*.msg()
print time() + *alert*.msg()

5.7.4 Implementasi *Blocker*

Blocker yang digunakan adalah *firewall*. Blocker ini dapat melakukan *block* packet pada ipv4 dan ipv6. Setiap paket yang berasal dari *host* yang dideteksi sebagai penyerang akan diblokir pada bagian *forward, input* dan *output*. Apabila IDS tersebut dimatikan, maka *rules* pada *iptable*s akan dihapus semua. *Psudocode* proses ini dapat dilihat pada Tabel 5.21.

Tabel 5.21 *Pseudocode* Blocker

Psudocode Blocker

If IPv4
  Table =*loadIptable*s()
else IPv6
  table =*load*Ip6tables()
rule = new *firewall*.rule()
rule.method = filter
rule.action = drop
rule.protocol = any
rule.chain = [input, output, forward]
table.insert(rule)
if IDS is exit
  table.flush()

5.7.5 Implementasi *Network Sniffer*

Sniffer akan menangkap semua paket pada *interface* yang dilindungi. *Sniffer* ini akan mengirimkan paket yang berhasil ditangkap kepada *detection engine*. *Pseudocode* dari *network sniffer* ini dapat dilihat pada Tabel 5.22.

Tabel 5.22 *Pseudocode Sniffer*

Psudocode Sniffer
pkts =Listen(iface) While pkts is receiving; do   Detect(pkts.pop())

BAB 6 PENGUJIAN DAN ANALISIS

6.1 Pengujian Performa IDS

Pengujian performa IDS dilakukan untuk mengukur dampak dari implementasi IDS pada *middleware*. Pengujian ini meliputi pengujian CPU, *memory* dan kecepatan dalam mengklasifikasikan paket.

6.1.1 Pengujian Penggunaan CPU Pada *Middleware*

Pengujian penggunaan CPU dilakukan untuk mengukur dampak dari implementasi IDS pada *middleware*. Pengujian ini dilakukan menghitung penggunaan CPU dalam waktu 10 menit. Haisl pengujian akan dibandingkan dengan ketika tidak menggunakan IDS.

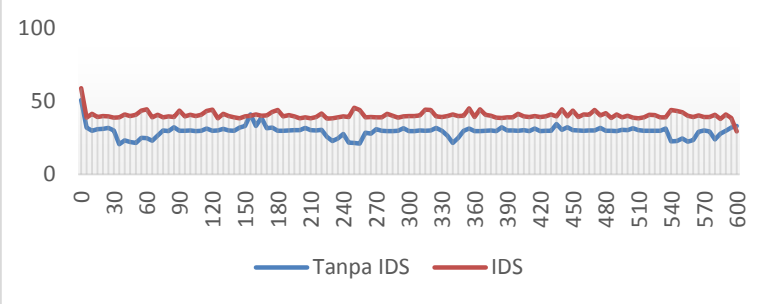
6.1.1.1 Pengujian Penggunaan CPU pada Trafik Normal

Pengujian ini dilakukan untuk mengukur penggunaan CPU dari implementasi IDS pada *middleware* ketika trafik sedang normal dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.1.

Tabel 6.1 Pengujian Penggunaan CPU pada Trafik Normal

Kode	KC_01
Pengujian	Penggunaan CPU pada Trafik Normal
Tujuan	Mengetahui penggunaan CPU pada *middleware* ketika trafik normal
Prosedur	1. Menjalankan IDS pada *middleware* 2. Merekam CPU pada trafik normal selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Melakukan perhitungan CPU
Hasil Yang diharapkan	Dapat mengetahui rata-rata penggunaan CPU pada trafik normal
Hasil Pengujian	Mengetahui rata-rata penggunaan CPU pada trafik normal

Rata-rata penggunaan CPU pada trafik normal tanpa IDS 29,3% sedangkan ketika menggunakan IDS 40,48%. Hasil ini dapat dilihat pada Gambar 6.1.



Gambar 6.1 Hasil Penggunaan CPU pada Trafik Normal

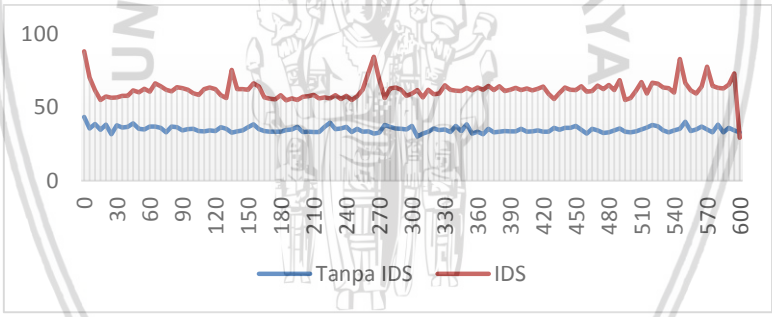
6.1.1.2 Pengujian Penggunaan *CPU* pada Serangan *UDP flood IPv4*

Pengujian ini dilakukan untuk mengukur penggunaan CPU dari implementasi IDS pada *middleware* ketika trafik terjadi serangan UDP *flood* IPv4 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.2.

Tabel 6.2 Pengujian penggunaan CPU pada serangan UDP *Food IPv4*

Kode	KC_02
Pengujian	Penggunaan CPU pada Serangan UDP *flood* IPv4
Tujuan	Mengetahui penggunaan CPU pada Serangan UDP *flood* IPv4
Prosedur	1. Menjalankan IDS pada *middleware* 2. Merekam CPU pada selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan UDP *flood* IPv4 5. Melakukan perhitungan CPU
Hasil Yang diharapkan	Dapat mengetahui rata-rata penggunaan CPU pada serangan UDP *flood* IPv4
Hasil Pengujian	Mengetahui rata-rata penggunaan CPU pada serangan UDP *flood* IPv4

Rata-rata penggunaan CPU tanpa IDS 35% sedangkan ketika menggunakan IDS 62,3%. Hasil penggunaan CPU dapat dilihat pada Gambar 6.2.



Gambar 6.2 Hasil Penggunaan CPU pada *pada Serangan* UDP *flood* IPv4

6.1.1.3 Pengujian Penggunaan *CPU* pada Serangan *UDP flood IPv6*

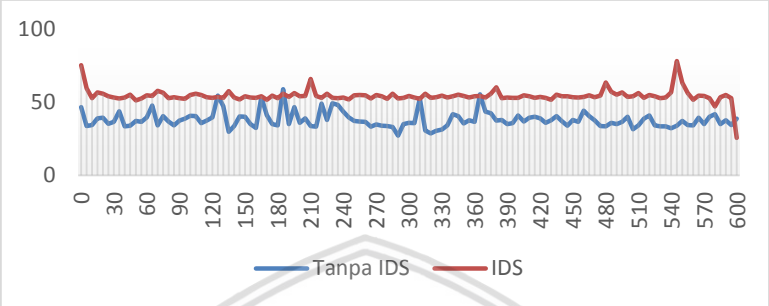
Pengujian ini dilakukan untuk mengukur penggunaan CPU dari implementasi IDS pada *middleware* ketika trafik terjadi serangan UDP *flood* IPv6 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.3.

Tabel 6.3 Pengujian penggunaan CPU pada serangan UDP *floowd IPv6*

Kode	KC_03
Pengujian	Penggunaan CPU pada Serangan UDP *flood* IPv6
Tujuan	Mengetahui penggunaan CPU pada serangan UDP *flood* IPv6
Prosedur	1. Menjalankan IDS pada *middleware* 2. Merekam CPU selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan UDP *flood* IPv6 5. Melakukan perhitungan CPU

Hasil Yang diharapkan	Dapat mengetahui rata-rata penggunaan CPU pada serangan UDP *flood* IPv6
Hasil Pengujian	Mengetahui rata-rata penggunaan CPU pada serangan UDP *flood* IPv6

Rata-rata penggunaan CPU tanpa IDS 38% sedangkan ketika menggunakan IDS 54,7%. Hasil penggunaan CPU dapat dilihat pada Gambar 6.3.



Gambar 6.3 Hasil Penggunaan CPU pada *pada Serangan* SYN *flood* IPv4

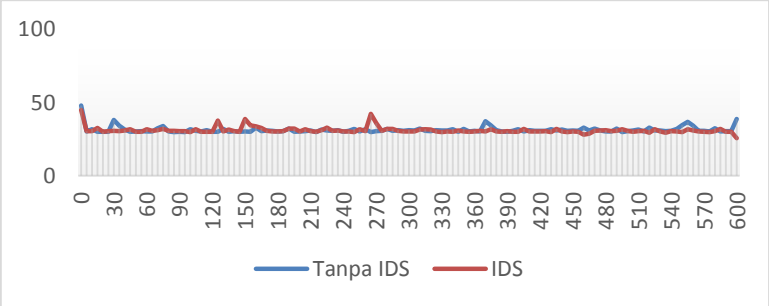
6.1.1.4 Pengujian Penggunaan *CPU* pada Serangan *SYN flood IPv4*

Pengujian ini dilakukan untuk mengukur penggunaan CPU dari implementasi IDS pada *middleware* ketika trafik terjadi serangan SYN *flood* IPv4 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.4.

Tabel 6.4 Penggunaan CPU pada serangan SYN *flood* IPv4

Kode	KC_04
Pengujian	Penggunaan CPU pada Serangan SYN *flood* IPv4
Tujuan	Mengetahui penggunaan CPU pada Serangan SYN *flood* IPv4
Prosedur	1. Menjalankan IDS pada *middleware* 2. Merekam CPU selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan SYN *flood* IPv4 5. Melakukan perhitungan CPU
Hasil Yang diharapkan	Dapat mengetahui rata-rata penggunaan CPU pada Serangan SYN *flood* IPv4
Hasil Pengujian	Mengetahui rata-rata penggunaan CPU pada Serangan SYN *flood* IPv4

Rata-rata penggunaan CPU tanpa IDS 3,.3% sedangkan ketika menggunakan IDS 31,1%. Hasil penggunaan CPU dapat dilihat pada Gambar 6.4.



Gambar 6.4 Hasil Penggunaan CPU pada *pada Serangan* SYN *flood* IPv4

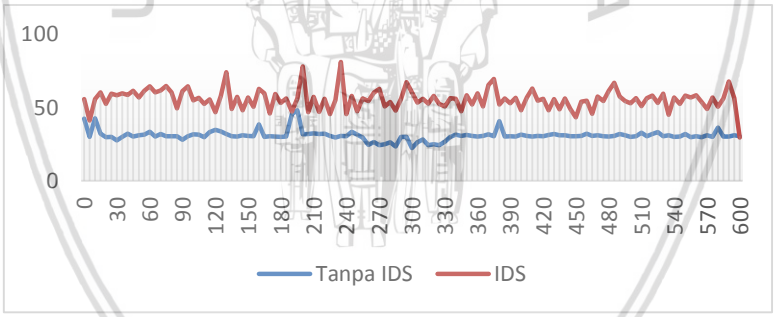
6.1.1.5 Pengujian Penggunaan *CPU* pada Serangan *SYN flood IPv6*

Pengujian ini dilakukan untuk mengukur penggunaan CPU dari implementasi IDS pada *middleware* ketika trafik terjadi serangan SYN *flood* IPv6 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.5.

Tabel 6.5 Penggunaan CPU pada serangan SYN *flood* IPv6

Kode	KC_05
Pengujian	Penggunaan CPU pada Serangan SYN *flood* IPv6
Tujuan	Mengetahui penggunaan CPU pada Serangan SYN *flood* IPv6
Prosedur	1. Menjalankan IDS pada *middleware* 2. Merekam CPU selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan SYN *flood* IPv6 5. Melakukan perhitungan CPU
Hasil Yang diharapkan	Dapat mengetahui rata-rata penggunaan CPU pada Serangan SYN *flood* IPv6
Hasil Pengujian	Mengetahui rata-rata penggunaan CPU pada Serangan SYN *flood* IPv6

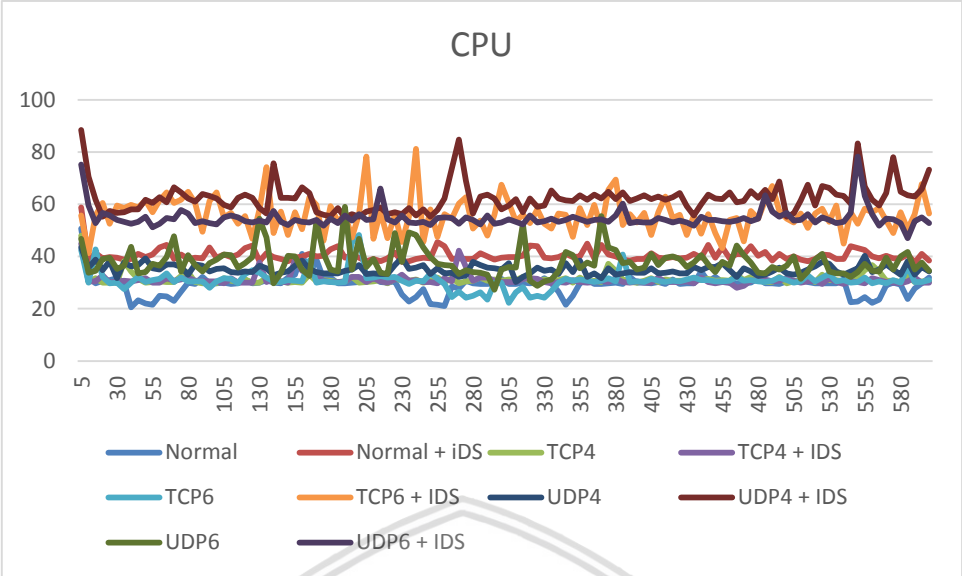
Rata-rata penggunaan CPU tanpa IDS 31% sedangkan ketika menggunakan IDS 55,9%. Hasil penggunaan CPU dapat dilihat pada Gambar 6.5.



Gambar 6.5 Hasil Penggunaan CPU pada *pada Serangan* SYN *flood* IPv6

6.1.1.6 Hasil Pengujian Performa IDS

Pengujian CPU dilakukan untuk mengetahui perbandingan penggunaan CPU ketika menggunakan IDS dan tidak menggunakan IDS. Pengujian ini dilakukan pada trafik serangan dan trafik normal. Hasil pengujian menunjukkan peningkatan rata-rata sebesar 16%. Hasil pengujian CPU secara keseluruhan dapat dilihat pada Gambar 6.6.



Gambar 6.6 Hasil Pengujian CPU

6.1.2 Pengujian Penggunaan *Memory* Pada *Middleware*

Pengujian penggunaan *Memory* dilakukan untuk mengukur dampak dari implementasi IDS pada *middleware*. Pengujian ini dilakukan menghitung penggunaan *Memory* dalam waktu 10 menit. Haisl pengujian akan dibandingkan dengan ketika tidak menggunakan IDS

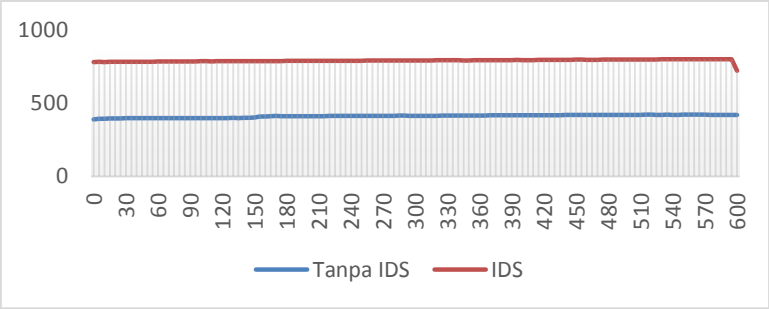
6.1.2.1 Pengujian Penggunaan *Memory pada Trafik Normal*

Pengujian ini dilakukan untuk mengukur penggunaan CPU dari implementasi IDS pada *middleware* ketika trafik sedang normal dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.6.

Tabel 6.6 Penggunaan *memory* pada trafik normal

Kode	KM_01
Pengujian	Penggunaan *memory* pada Trafik Normal
Tujuan	Mengetahui penggunaan *memory* pada Trafik Normal
Prosedur	1. Menjalankan IDS pada *middleware* 2. Merekam *memory* pada trafik normal selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Melakukan perhitungan *memory*
Hasil Yang diharapkan	Dapat mengetahui rata-rata penggunaan *memory* pada Trafik Normal
Hasil Pengujian	Mengetahui rata-rata penggunaan *memory* pada Trafik Normal

Rata-rata penggunaan *memory* pada trafik normal tanpa IDS 413MB sedangkan ketika menggunakan IDS 793MB. Hasil penggunaan *memory* dapat dilihat pada Gambar 6.7.



Gambar 6.7 Hasil Penggunaan *Memory* pada trafik normal

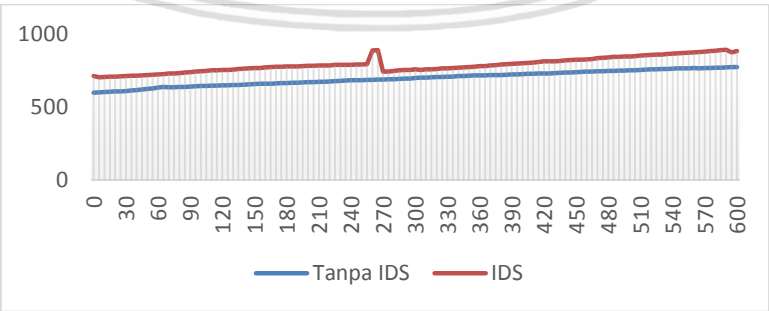
6.1.2.2 Pengujian Penggunaan *memory* pada Serangan *UDP flood IPv4*

Pengujian ini dilakukan untuk mengukur penggunaan CPU dari implementasi IDS pada *middleware* ketika trafik terjadi serangan UDP *flood* IPv4 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.7.

Tabel 6.7 Pengujian penggunaan *memory* pada serangan UDP *flood IPv4*

Kode	KM_02
Pengujian	Penggunaan *memory* pada Serangan UDP *flood* IPv4
Tujuan	Mengetahui penggunaan *memory* pada Serangan UDP *flood* IPv4
Prosedur	1. Menjalankan IDS pada *middleware* 2. Merekam *memory* selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan UDP *flood* IPv4 5. Melakukan perhitungan *memory*
Hasil Yang diharapkan	Dapat mengetahui rata-rata penggunaan *memory* pada serangan UDP *flood* IPv4
Hasil Pengujian	Mengetahui rata-rata penggunaan *memory* pada serangan UDP *flood* IPv4

Rata-rata penggunaan *memory* tanpa IDS 694 MB sedangkan ketika menggunakan IDS 788 MB. Hasil penggunaan *memory* dapat dilihat pada Gambar 6.8.



Gambar 6.8 Hasil Penggunaan *Memory* pada *pada Serangan* UDP *flood* IPv4

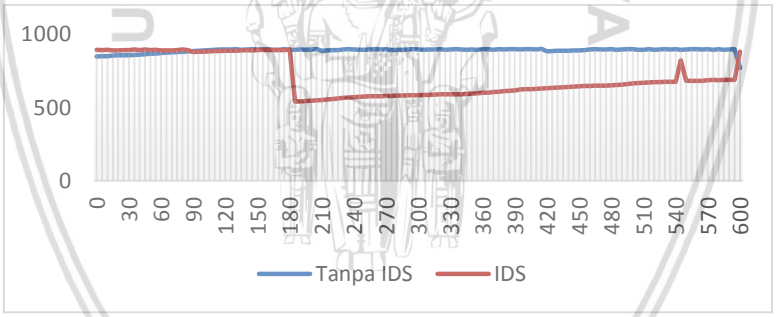
6.1.2.3 Pengujian Penggunaan *memory* pada Serangan *UDP flood IPv6*

Pengujian ini dilakukan untuk mengukur penggunaan *memory* dari implementasi IDS pada *middleware* ketika trafik terjadi serangan UDP *flood* IPv6 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.8.

Tabel 6.8 Pengujian penggunaan *memory* pada serangan UDP *flood IPv6*

Kode	KM_03
Pengujian	Penggunaan *memory* pada Serangan UDP *flood* IPv6
Tujuan	Mengetahui penggunaan *memory* pada serangan UDP *flood* IPv6
Prosedur	1. Menjalankan IDS pada *middleware* 2. Merekam *memory* selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan UDP *flood* IPv6 5. Melakukan perhitungan *memory*
Hasil Yang diharapkan	Dapat mengetahui rata-rata penggunaan *memory* pada serangan UDP *flood* IPv6
Hasil Pengujian	Mengetahui rata-rata penggunaan *memory* pada serangan UDP *flood* IPv6

Rata-rata penggunaan *memory* tanpa IDS 891 MB sedangkan ketika menggunakan IDS 704 MB. Hasil penggunaan *memory* dapat dilihat pada Gambar 6.9.



Gambar 6.9 Hasil Penggunaan *Memory* pada *pada Serangan* UDP *flood* IPv6

6.1.2.4 Pengujian Penggunaan *memory* pada Serangan *SYN flood IPv4*

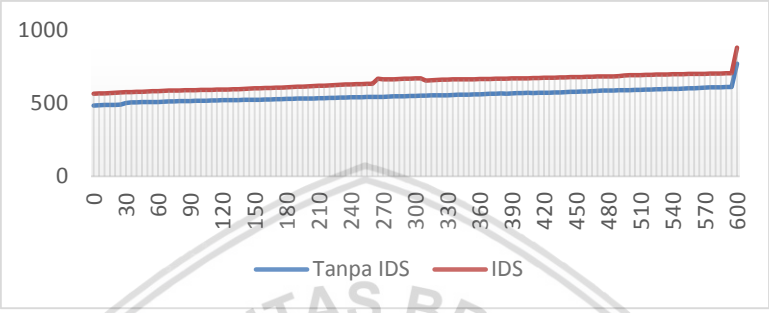
Pengujian ini dilakukan untuk mengukur penggunaan *memory* dari implementasi IDS pada *middleware* ketika trafik terjadi serangan SYN *flood* IPv4 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.9.

Tabel 6.9 Penggunaan CPU pada serangan SYN *flood* IPv4

Kode	KM_04
Pengujian	Penggunaan *memory* pada Serangan SYN *flood* IPv4
Tujuan	Mengetahui penggunaan *memory* pada Serangan SYN *flood* IPv4
Prosedur	1. Menjalankan IDS pada *middleware* 2. Merekam *memory* selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan SYN *flood* IPv4

	5. Melakukan perhitungan *memory*
Hasil Yang diharapkan	Dapat mengetahui rata-rata penggunaan *memory* pada Serangan SYN *flood* IPv4
Hasil Pengujian	Mengetahui rata-rata penggunaan *memory* pada Serangan SYN *flood* IPv4

Rata-rata penggunaan *memory* tanpa IDS 550 MB sedangkan ketika menggunakan IDS 643 MB. Hasil penggunaan *memory* dapat dilihat pada Gambar 6.10.



Gambar 6.10 Hasil Penggunaan *Memory* pada *pada Serangan* SYN *flood* IPv4

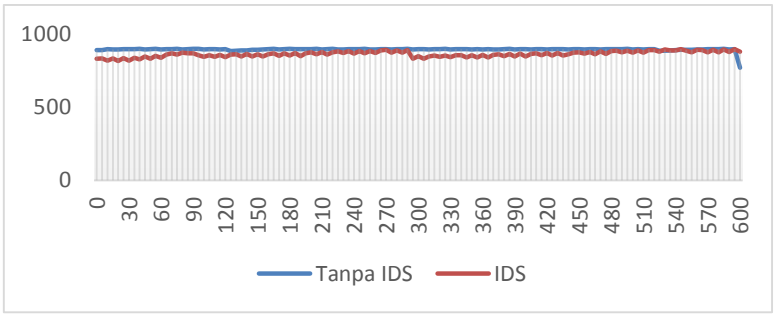
6.1.2.5 Pengujian Penggunaan *memory* pada Serangan *SYN flood IPv6*

Pengujian ini dilakukan untuk mengukur penggunaan *memory* dari implementasi IDS pada *middleware* ketika trafik terjadi serangan SYN *flood* IPv6 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.10.

Tabel 6.10 Penggunaan CPU pada serangan SYN *flood* IPv6

Kode	KM_05
Pengujian	Penggunaan *memory* pada Serangan SYN *flood* IPv6
Tujuan	Mengetahui penggunaan *memory* pada Serangan SYN *flood* IPv6
Prosedur	1. Menjalankan IDS pada *middleware* 2. Merekam *memory* selama 10 menit pada *middleware* 3. Sensor mengirimkan data setiap satu menit 4. Mengirimkan trafik serangan SYN *flood* IPv6 5. Melakukan perhitungan *memory*
Hasil Yang diharapkan	Dapat mengetahui rata-rata penggunaan *memory* pada Serangan SYN *flood* IPv6
Hasil Pengujian	Mengetahui rata-rata penggunaan *memory* pada Serangan SYN *flood* IPv6

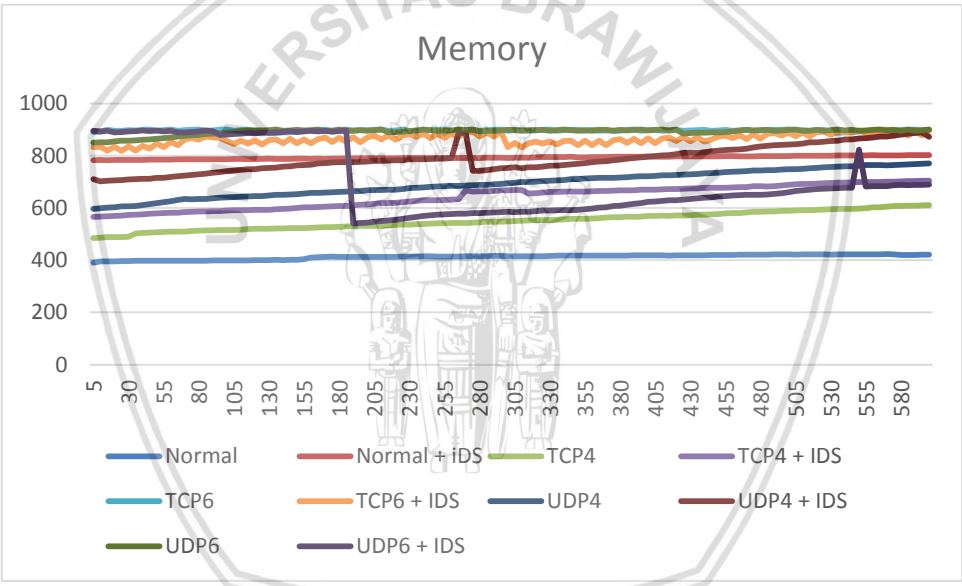
Rata-rata penggunaan *memory* tanpa IDS 897 MB sedangkan ketika menggunakan IDS 864 MB. Hasil penggunaan *memory* dapat dilihat pada Gambar 6.11.



Gambar 6.11 Hasil Penggunaan *Memory* pada *pada Serangan* SYN *flood* IPv6

6.1.2.6 Hasil Pengujian Performa IDS

Pengujian *memory* dilakukan dengan mengetahui perbandingan penggunaan *memory* ketika menggunakan IDS dan tidak menggunakan IDS. Pengujian ini dilakukan pada trafik serangan dan trafik normal. Hasil pengujian menunjukkan peningkatan rata-rata sebesar 70mb. Hasil pengujian *memory* secara keseluruhan dapat dilihat pada Gambar 6.12.



Gambar 6.12 Hasil Pengujian *Memory*

6.1.3 Pengujian Kecepatan Klasifikasi Paket

Pengujian kecepatan dilakukan untuk menghitung kecepatan IDS untuk mengklasifikasikan paket dalam waktu 10 menit.

6.1.3.1 Pengujian Kecepatan Klasifikasi *pada Trafik Normal*

Pengujian ini dilakukan untuk mengukur kecepatan klasifikasi dari implementasi IDS pada *middleware* ketika trafik sedang normal dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.11.

Tabel 6.11 Pengujian kecepatan klasifikasi pada trafik normal

Kode	KK_01

Pengujian	Kecepatan klasifikasi pada Trafik Normal
Tujuan	Mengetahui kecepatan klasifikasi pada Trafik Normal
Prosedur	1. Menjalankan IDS pada *middleware* 2. Menghitung waktu proses paket pada trafik normal selama 10 menit pada *middleware* 3. Melakukan perhitungan rata-rata kecepatan klasifikasi paket
Hasil Yang diharapkan	Dapat mengetahui rata-rata kecepatan klasifikasi pada Trafik Normal
Hasil Pengujian	Mengetahui rata-rata kecepatan klasifikasi pada Trafik Normal

Pengujian ini dilakukan dengan merekam kecepatan klasifikasi ketika IDS berjalan. Hasil kecepatan klasifikasi setelah dihitung rata-ratanya adalah 0,00053476 detik.

6.1.3.2 Pengujian Kecepatan Klasifikasi pada Serangan *UDP flood IPv4*

Pengujian ini dilakukan untuk mengukur kecepatan klasifikasi dari implementasi IDS pada *middleware* ketika trafik terjadi serangan UDP *flood* IPv4 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.12.

Tabel 6.12 Pengujian kecepatan klasifikasi pada serangan UDP *flood IPv4*

Kode	KK_02
Pengujian	Kecepatan klasifikasi pada Serangan UDP *flood* IPv4
Tujuan	Mengetahui kecepatan klasifikasi pada Serangan UDP *flood* IPv4
Prosedur	1. Menjalankan IDS pada *middleware* 2. Menghitung waktu proses paket pada trafik serangan UDP *flood* IPv4 selama 10 menit pada *middleware* 3. Melakukan perhitungan rata-rata kecepatan klasifikasi paket
Hasil Yang diharapkan	Dapat mengetahui rata-rata kecepatan klasifikasi pada serangan UDP *flood* IPv4
Hasil Pengujian	Mengetahui rata-rata kecepatan klasifikasi pada serangan UDP *flood* IPv4

Pengujian ini dilakukan dengan merekam kecepatan klasifikasi ketika IDS berjalan. Hasil kecepatan klasifikasi setelah dihitung rata-ratanya adalah 0,03532629 detik.

6.1.3.3 Pengujian Kecepatan Klasifikasi pada Serangan *UDP flood IPv6*

Pengujian ini dilakukan untuk mengukur kecepatan klasifikasi dari implementasi IDS pada *middleware* ketika trafik terjadi serangan UDP *flood* IPv6 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.13.

Tabel 6.13 Pengujian kecepatan klasifikasi pada serangan UDP *flood IPv6*

Kode	KK_03
Pengujian	Kecepatan klasifikasi pada Serangan UDP *flood* IPv6
Tujuan	Mengetahui kecepatan klasifikasi pada serangan UDP *flood* IPv6

Prosedur	1. Menjalankan IDS pada *middleware* 2. Menghitung waktu proses paket pada trafik serangan UDP *flood* IPv6 selama 10 menit pada *middleware* 3. Melakukan perhitungan rata-rata kecepatan klasifikasi paket
Hasil Yang diharapkan	Dapat mengetahui rata-rata kecepatan klasifikasi pada serangan UDP *flood* IPv6
Hasil Pengujian	Mengetahui rata-rata kecepatan klasifikasi pada serangan UDP *flood* IPv6

Pengujian ini dilakukan dengan merekam kecepatan klasifikasi ketika IDS berjalan. Hasil kecepatan klasifikasi setelah dihitung rata-ratanya adalah 0, 03954160 detik.

6.1.3.4 Pengujian Kecepatan Klasifikasi pada Serangan *SYN flood IPv4*

Pengujian ini dilakukan untuk mengukur kecepatan klasifikasi dari implementasi IDS pada *middleware* ketika trafik terjadi serangan SYN *flood* IPv4 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.14.

Tabel 6.14 Kecepatan klasifikasi pada serangan SYN *flood* IPv4

Kode	KK_04
Pengujian	Kecepatan klasifikasi pada Serangan SYN *flood* IPv4
Tujuan	Mengetahui kecepatan klasifikasi pada Serangan SYN *flood* IPv4
Prosedur	1. Menjalankan IDS pada *middleware* 2. Menghitung waktu proses paket pada trafik serangan SYN *flood* IPv4 selama 10 menit pada *middleware* 3. Melakukan perhitungan rata-rata kecepatan klasifikasi paket
Hasil Yang diharapkan	Dapat mengetahui rata-rata kecepatan klasifikasi pada Serangan SYN *flood* IPv4
Hasil Pengujian	Mengetahui rata-rata kecepatan klasifikasi pada Serangan SYN *flood* IPv4

Pengujian ini dilakukan dengan merekam kecepatan klasifikasi ketika IDS berjalan. Hasil kecepatan klasifikasi setelah dihitung rata-ratanya adalah 0,03737132detik.

6.1.3.5 Pengujian Kecepatan Klasifikasi pada Serangan *SYN flood IPv6*

Pengujian ini dilakukan untuk mengukur kecepatan klasifikasi dari implementasi IDS pada *middleware* ketika trafik terjadi serangan SYN *flood* IPv6 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.15.

Tabel 6.15 Kecepatan klasifikasi pada serangan SYN *flood* IPv6

Kode	KK_05
Pengujian	Kecepatan klasifikasi pada Serangan SYN *flood* IPv6
Tujuan	Mengetahui kecepatan klasifikasi pada Serangan SYN *flood* IPv6
Prosedur	1. Menjalankan IDS pada *middleware* 2. Menghitung waktu proses paket pada trafik serangan SYN *flood* IPv6 selama 10 menit pada *middleware*

	3. Melakukan perhitungan rata-rata kecepatan klasifikasi paket
Hasil Yang diharapkan	Dapat mengetahui rata-rata kecepatan klasifikasi pada Serangan SYN *flood* IPv6
Hasil Pengujian	Mengetahui rata-rata kecepatan klasifikasi pada Serangan SYN *flood* IPv6

Pengujian ini dilakukan dengan merekam kecepatan klasifikasi ketika IDS berjalan. Hasil kecepatan klasifikasi setelah dihitung rata-ratanya adalah 0,04037132 detik.

6.1.3.6 Hasil Kecepatan Klasifikasi Paket

Hasil rata-rata yang didapatkan untuk pengujian ini adalah 0,030629 detik Hasil dari rata-rata kecepatan klasifikasi pada Tabel 6.16.

Tabel 6.16 Rata-rata Kecepatan Klasifikasi (detik)

Kategori	**Rata-rata**
NORMAL	0,000535
SYN *FLOOD* IPv4	0,03737
SYN *FLOOD* IPv6	0,040371
UDP *FLOOD* IPv4	0,035326
UDP *FLOOD* IPv6	0,039542
Rata-rata	0.030629

6.2 Pengujian Fungsional IDS

Pengujian Fungsionalitas adalah pengujian yang dilakukan untuk menentukkan keberhasilan dari setiap fungsionalitas yang telah dirancang. Pengujian pada bagian ini menggunakan metode *black box*. Metode ini akan melakukan validasi pada setiap fungsionalitas berdasarkan input dan output yang dihasilkan.

6.2.1 Pengujian Akurasi Pengambilan Trafik IDS

Pengujian akurasi pengambilan trafik digunakan untuk menguji kesesuaian antara paket yang diterima pada IDS dengan tcpdump. Pengujian ini dilakukan pada saat trafik normal dan trafik serangan.

File yang dikumpulkan dalam proses pengujian akan dikumpulkan dan ditammpilkan dengan *script bash*. Adapun psudocode dari *script* tersebut dapat dilihat pada Tabel 6.17.

Tabel 6.17 Psuedocode Ekstraksi Akurasi Scapy

Psuedocode

for each pcapfiles in directory as file; do
  scapy = read file.filename() + 'txt' | grep 'received' | countEachLine
  tcpdump = read file | countEachLine
  echo file.filename() + 'txt scapy:' + scapy
  echo file.filename() + 'pcap tcpdump' + tcpdump
endfor

6.2.1.1 Pengujian Trafik Normal

Trafik normal adalah trafik yang ada pada lingkungan penelitian. Pengujian akurasi dilakukan untuk mendapatkan jumlah paket yang diterima dengan IDS. Hasil dari proses pengujian ini dapat dilhat pada Tabel 6.18.

Tabel 6.18 Pengujian Trafik Normal

Kode	AT_01
Pengujian	Trafik Normal
Tujuan	Mengetahui akurasi trafik yang dapat diambil pada IDS
Prosedur	1. IDS diset untuk menampilkan pesan setiap ada paket yang masuk 2. Sensor mengirimkan paket setiap satu menit 3. IDS dan tcpdump dijalankan di *middleware* secara bersamaan selama lima menit 4. Jumlah pesan paket masuk pada IDS dihitung 5. Menghitung paket masuk pada IDS 6. Melakukan komparasi antara hasil dari IDS dengan tcpdump
Hasil Yang diharapkan	Daoat mengetahui akurasi trafik yang dapat diambil pada IDS
Hasil Pengujian	Akurasi dari IDS ketika merekam trafik normal sebesar 96.73%

Pada trafik normal, scapy memiliki kesesuaian paket rata-rata sebesar 96,73% dibandingkan dengan tcpdump. Pada Tabel 6.19 menunjukkan hasil dari pengujian AT_01.

Tabel 6.19 Akurasi Scapy Pada Trafik Normal

No	Scapy	Tcpdump	Akurasi
1	10277	10626	96,72%
2	10271	10589	97,00%
3	10466	10797	96,93%
4	10075	10470	96,23%
5	10573	10930	96,73%
6	10277	10631	96,67%
7	10375	10715	96,83%
8	10077	10485	96,11%
9	10373	10693	97,01%
10	10380	10695	97,05%
Rata-rata			96,73%

6.2.1.2 Pengujian Trafik Serangan UDP Flood IPv4

Trafik Serangan UDP *Flood* IPv4 adalah trafik yang ada pada lingkungan penelitian ketika terjadi serangan UDP *flood* IPv4. Pengujian akurasi dilakukan untuk mendapatkan jumlah paket yang diterima dengan IDS. Hasil dari proses pengujian ini dapat dilhat pada Tabel 6.20.

Tabel 6.20 Pengujian Trafik Serangan UDP *Flood* IPv4

Kode	AT_02
Pengujian	Trafik Serangan UDP *Flood* IPv4
Tujuan	Mengetahui akurasi trafik yang dapat diambil pada IDS ketika terjadi serangan
Prosedur	1. IDS diset untuk menampilkan pesan setiap ada paket yang masuk 2. Sensor mengirimkan paket setiap satu menit 3. IDS dan tcpdump dijalankan di *middleware* secara bersamaan selama lima menit 4. *Middleware* diserang denga UDP *flood* IPv4 5. Jumlah pesan paket masuk pada IDS dihitung 6. Menghitung paket masuk pada IDS 7. Melakukan komparasi antara hasil dari IDS dengan tcpdump
Hasil Yang diharapkan	Daoat mengetahui akurasi trafik yang dapat diambil pada IDS ketika terjadi serangan
Hasil Pengujian	Akurasi dari IDS ketika merekam trafik serangan UDP *flood* IPv4 sebesar 64.86%

Pada trafik serangan UDP *flood* IPv4, scapy memiliki kesesuaian paket rata-rata sebesar 64,86% dibandingkan dengan tcpdump. Pada Tabel 6.21 menunjukkan hasil dari pengujian AT_02.

Tabel 6.21 Akurasi Scapy Pada Trafik UDP *flood* IPv4

No	Scapy	Tcpdump	Akurasi
1	11837	19695	60,10%
2	11415	18828	60,63%
3	13248	19425	68,20%
4	14172	19484	72,74%
5	11762	19000	61,91%
6	12628	19397	65,10%
7	14460	19292	74,95%
8	11663	18947	61,56%
9	12313	19037	64,68%
10	11438	19473	58,74%
Rata-rata			64,86%

6.2.1.3 Pengujian Trafik Serangan UDP Flood IPv6

Trafik Serangan UDP *Flood* IPv6 adalah trafik yang ada pada lingkungan penelitian ketika terjadi serangan UDP *flood* IPv6. Pengujian akurasi dilakukan untuk mendapatkan jumlah paket yang diterima dengan IDS. Hasil dari proses pengujian ini dapat dilhat pada Tabel 6.22.

Tabel 6.22 Pengujian Trafik Serangan UDP *Flood* IPv6

Kode	AT_03
Pengujian	Trafik Serangan UDP *Flood* IPv6

Tujuan	Mengetahui akurasi trafik yang dapat diambil pada IDS ketika terjadi serangan
Prosedur	1. IDS diset untuk menampilkan pesan setiap ada paket yang masuk 2. Sensor mengirimkan paket setiap satu menit 3. IDS dan tcpdump dijalankan di *middleware* secara bersamaan selama lima menit 4. *Middleware* diserang denga UDP *flood* IPv6 5. Jumlah pesan paket masuk pada IDS dihitung 6. Menghitung paket masuk pada IDS 7. Melakukan komparasi antara hasil dari IDS dengan tcpdump
Hasil Yang diharapkan	Daoat mengetahui akurasi trafik yang dapat diambil pada IDS ketika terjadi serangan
Hasil Pengujian	Akurasi dari IDS ketika merekam trafik serangan UDP *flood* IPv6 sebesar 64.86%

Pada trafik serangan UDP *flood* IPv6, scapy memiliki kesesuaian paket rata-rata sebesar 68,02% dibandingkan dengan tcpdump. Pada Tabel 6.23 menunjukkan hasil dari pengujian AT_03.

Tabel 6.23 Akurasi Scapy Pada Trafik UDP IPv6

No	Scapy	Tcpdump	Akurasi
1	13153	20594	63,87%
2	15374	20840	73,77%
3	16411	21107	77,75%
4	16298	20603	79,10%
5	13908	21064	66,03%
6	11602	19096	60,76%
7	13248	19337	68,51%
8	12758	19425	65,68%
9	13223	19527	67,72%
10	14698	19802	74,22%
Rata-rata			69,74%

6.2.1.4 Pengujian Trafik Serangan SYN flood IPv4

Trafik Serangan SYN *Flood* IPv4 adalah trafik yang ada pada lingkungan penelitian ketika terjadi serangan SYN *flood* IPv4. Pengujian akurasi dilakukan untuk mendapatkan jumlah paket yang diterima dengan IDS. Hasil dari proses pengujian ini dapat dilhat pada Tabel 6.24.

Tabel 6.24 Pengujian Trafik Serangan SYN *flood* IPv4

Kode	AT_04
Pengujian	Trafik Serangan SYN *flood* IPv4
Tujuan	Mengetahui akurasi trafik yang dapat diambil pada IDS ketika terjadi serangan
Prosedur	1. IDS diset untuk menampilkan pesan setiap ada paket yang masuk 2. Sensor mengirimkan paket setiap satu menit

	3. IDS dan tcpdump dijalankan di *middleware* secara bersamaan selama lima menit 4. *Middleware* diserang denga SYN *flood* IPv4 5. Jumlah pesan paket masuk pada IDS dihitung 6. Menghitung paket masuk pada IDS 7. Melakukan komparasi antara hasil dari IDS dengan tcpdumpkomparasi antara hasil dari IDS dengan tcpdump
Hasil Yang diharapkan	Daoat mengetahui akurasi trafik yang dapat diambil pada IDS ketika terjadi serangan
Hasil Pengujian	Akurasi dari IDS ketika merekam trafik serangan SYN *flood* IPv4 sebesar 64.86%

Pada trafik serangan SYN *flood* IPv4, scapy memiliki kesesuaian paket rata-rata sebesar 68,02% dibandingkan dengan tcpdump. Pada Tabel 6.25 menunjukkan hasil dari pengujian AT_04.

Tabel 6.25 Pengujian Akurasi Scapy Pada Trafik SYN *flood* IPv4

No	Scapy	Tcpdump	Akurasi
1	15008	21722	69,09%
2	15619	21796	71,66%
3	13917	21180	65,71%
4	13012	21196	61,39%
5	14401	21278	67,68%
6	15480	21840	70,88%
7	15271	21501	71,02%
8	12589	21237	59,28%
9	14886	21325	69,81%
10	15831	21498	73,64%
Rata-rata			68,02%

6.2.1.5 Pengujian Trafik Serangan SYN flood IPv6

Trafik Serangan SYN *Flood* IPv6 adalah trafik yang ada pada lingkungan penelitian ketika terjadi serangan SYN *flood* IPv6. Pengujian akurasi dilakukan untuk mendapatkan jumlah paket yang diterima dengan IDS. Hasil dari proses pengujian ini dapat dilhat pada Tabel 6.26.

Tabel 6.26 Pengujian Akurasi Scapy Pada Trafik Serangan SYN *flood* IPv6

Kode	AT_05
Pengujian	Trafik Serangan SYN *flood* IPv6
Tujuan	Mengetahui akurasi trafik yang dapat diambil pada IDS ketika terjadi serangan
Prosedur	1. IDS diset untuk menampilkan pesan setiap ada paket yang masuk 2. Sensor mengirimkan paket setiap satu menit 3. IDS dan tcpdump dijalankan di *middleware* secara bersamaan selama lima menit 4. *Middleware* diserang denga SYN *flood* IPv6 5. Jumlah pesan paket masuk pada IDS dihitung

	6. Menghitung paket masuk pada IDS 7. Melakukan komparasi antara hasil dari IDS dengan tcpdumpantara hasil dari IDS dengan tcpdump
Hasil Yang diharapkan	Daoat mengetahui akurasi trafik yang dapat diambil pada IDS ketika terjadi serangan
Hasil Pengujian	Akurasi dari IDS ketika merekam trafik serangan SYN *flood* IPv6 sebesar 64.86%

Pada trafik serangan SYN *flood* IPv6, scapy memiliki kesesuaian paket rata-rata sebesar 79,91% dibandingkan dengan tcpdump. Pada Tabel 6.27 menunjukkan hasil dari pengujian AT_05.

Tabel 6.27 Hasil Pengujian Akurasi Scapy Pada Trafik Serangan SYN *flood* IPv6

No	Scapy	Tcpdump	Akurasi
1	69.24%	21276	69,24%
2	62.21%	20569	62,21%
3	71.26%	20682	71,26%
4	60.80%	20472	60,80%
5	72.43%	20803	72,43%
6	73.63%	20664	73,63%
7	72.23%	12192	97,81%
8	69.22%	10799	97,63%
9	65.40%	10789	96,95%
10	70.13%	10761	97,17%
Rata-rata			68,66%

6.2.1.6 Hasil Pengujian Akurasi

Hasil rata-rata yang didapatkan untuk pengujian ini adalah 75,85%. Jumlah ini cukup untuk digunakan dalam pendeteksian serangan pada perangkat IoT. Sehingga IDS ini dapat tetap berjalan meskipun paket yang dapat diterima tidak dapat diterima semuanya. Hasil dari akurasi scapy dapat dilihat pada Tabel 6.28.

Tabel 6.28 Rata-rata Akurasi Scapy

Kategori	Rata-rata Akurasi
NORMAL	96,73%
SYN *FLOOD* IPv4	68,02%
SYN *FLOOD* IPv6	68,66%
UDP *FLOOD* IPv4	64,86%
UDP *FLOOD* IPv6	69,74%
Rata-rata	73,60%

6.2.2 Pengujian Akurasi *Detection Engine* IDS

6.2.2.1 Pengujian Akurasi *Detection Engine pada Trafik Normal*

Pengujian ini dilakukan untuk mengukur akurasi *detection engine* dari implementasi IDS pada *middleware* ketika trafik sedang normal dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.29.

Tabel 6.29 Pengujian akurasi *detection* pada trafik normal

Kode	DE_01
Pengujian	Akurasi *detection engine* pada Trafik Normal
Tujuan	Mengetahui akurasi *detection engine* pada Trafik Normal
Prosedur	1. IDS dijalankan dalam waktu 5 menit 2. Sensor mengirimkan data dalam waktu satu menit 3. Melihat summary dari paket yang diklasifikasikan oleh IDS
Hasil Yang diharapkan	Dapat mengetahui rata-rata akurasi *detection engine* pada Trafik Normal
Hasil Pengujian	Mengetahui rata-rata akurasi *detection engine* pada Trafik Normal

Pengujian ini dilakukan dengan merekam akurasi *detection engine* ketika kondisi trafik normal dan IDS berjalan. Hasil kecepatan klasifikasi dapat dilihat pada Tabel 6.30.

Tabel 6.30 Pengujian kemampuan menampilkan *alert* terhadap trafik normal

Pengujian	TP	TN	FN	FP	Akurasi
1	165303	0	0	0	100%
2	161122	0	0	0	100%
3	162010	0	0	0	100%
4	168703	0	0	0	100%
5	166173	0	0	0	100%
Rata-rata					100%

6.2.2.2 Pengujian Akurasi *Detection Engine* pada Serangan *UDP flood IPv4*

Pengujian ini dilakukan untuk mengukur akurasi *detection engine* dari implementasi IDS pada *middleware* ketika trafik terjadi serangan UDP *flood* IPv4 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.31.

Tabel 6.31 Pengujian akurasi *detection engine* pada serangan UDP *flood IPv4*

Kode	DE_02
Pengujian	Akurasi *detection engine* pada Serangan UDP *flood* IPv4
Tujuan	Mengetahui akurasi *detection engine* pada Serangan UDP *flood* IPv4
Prosedur	1. IDS dijalankan 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv4 paket

	3. IDS dimatikan dan diihat summary dari paket yang diklasifikasikan oleh IDS
Hasil Yang diharapkan	Dapat mengetahui rata-rata akurasi *detection engine* pada serangan UDP *flood* IPv4
Hasil Pengujian	Mengetahui rata-rata akurasi *detection engine* pada serangan UDP *flood* IPv4

Pengujian ini dilakukan dengan merekam akurasi *detection engine* ketika kondisi serangan UDP *flood IPv4* dan IDS berjalan. Hasil kecepatan klasifikasi dapat dilihat Tabel 6.32.

Tabel 6.32 Pengujian kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv4

Pengujian	TP	TN	FN	FP	Akurasi
1	165526	0	0	0	100%
2	161298	0	0	0	100%
3	162178	0	0	0	100%
4	168887	0	0	0	100%
5	166344	0	0	0	100%
Rata-rata					100%

6.2.2.3 Pengujian Akurasi *Detection Engine* pada Serangan *UDP flood IPv6*

Pengujian ini dilakukan untuk mengukur akurasi *detection engine* dari implementasi IDS pada *middleware* ketika trafik terjadi serangan UDP *flood* IPv6 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.33.

Tabel 6.33 Pengujian akurasi *detection engine* pada serangan UDP *flood IPv6*

Kode	DE_03
Pengujian	Akurasi *detection engine* pada Serangan UDP *flood* IPv6
Tujuan	Mengetahui akurasi *detection engine* pada serangan UDP *flood* IPv6
Prosedur	1. IDS dijalankan 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv6 paket 3. IDS dimatikan dan diihat summary dari paket yang diklasifikasikan oleh IDS
Hasil Yang diharapkan	Dapat mengetahui rata-rata akurasi *detection engine* pada serangan UDP *flood* IPv6
Hasil Pengujian	Mengetahui rata-rata akurasi *detection engine* pada serangan UDP *flood* IPv6

Dan hasil pengujian dapat dilihat pada Tabel 6.34.

Tabel 6.34 Pengujian kemampuan menampilkan *alert* terhadap Serangan UDP *flood* Ipv6

Pengujian	TP	TN	FN	FP	Akurasi
1	165479	0	0	0	100%
2	161295	0	0	0	100%
3	162170	0	0	0	100%

4	168875	0	0	0	100%
5	166346	0	0	0	100%
Rata-rata					100%

6.2.2.4 Pengujian Akurasi *Detection Engine* pada Serangan *SYN flood IPv4*

Pengujian ini dilakukan untuk mengukur akurasi *detection engine* dari implementasi IDS pada *middleware* ketika trafik terjadi serangan SYN *flood* IPv4 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.35.

Tabel 6.35 Pengujian akurasi *detection engine* pada serangan SYN *flood* IPv4

Kode	DE_04
Pengujian	Akurasi *detection engine* pada Serangan SYN *flood* IPv4
Tujuan	Mengetahui akurasi *detection engine* pada Serangan SYN *flood* IPv4
Prosedur	1. IDS dijalankan 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 TCP IPv4 paket 3. IDS dimatikan dan diihat summary dari paket yang diklasifikasikan oleh IDS
Hasil Yang diharapkan	Dapat mengetahui rata-rata akurasi *detection engine* pada Serangan SYN *flood* IPv4
Hasil Pengujian	Mengetahui rata-rata akurasi *detection engine* pada Serangan SYN *flood* IPv4

Dan hasil pengujian dapat dilihat pada Tabel 6.36.

Tabel 6.36 Pengujian kemampuan menampilkan *alert* terhadap Serangan SYN *flood* IPv4

Pengujian	TP	TN	FN	FP	Akurasi
1	165459	0	0	0	100%
2	161284	0	0	0	100%
3	162176	0	0	0	100%
4	168867	0	0	0	100%
5	166344	0	0	0	100%
Rata-rata					100%

6.2.2.5 Pengujian Akurasi *Detection Engine* pada Serangan *SYN flood IPv6*

Pengujian ini dilakukan untuk mengukur akurasi *detection engine* dari implementasi IDS pada *middleware* ketika trafik terjadi serangan SYN *flood* IPv6 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.37.

Tabel 6.37 Pengujian akurasi *detection engine* pada serangan SYN *flood* IPv6

Kode	DE_05
Pengujian	Akurasi *detection engine* pada Serangan SYN *flood* IPv6
Tujuan	Mengetahui akurasi *detection engine* pada Serangan SYN *flood* IPv6
Prosedur	1. IDS dijalankan 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 TCP IPv6 paket

	3. IDS dimatikan dan diihat summary dari paket yang diklasifikasikan oleh IDS
Hasil Yang diharapkan	Dapat mengetahui rata-rata akurasi *detection engine* pada Serangan SYN *flood* IPv6
Hasil Pengujian	Mengetahui rata-rata akurasi *detection engine* pada Serangan SYN *flood* IPv6

Dan hasil pengujian dapat dilihat pada Tabel 6.38.

Tabel 6.38 Pengujian kemampuan menampilkan *alert* terhadap Serangan SYN *flood* Ipv6

Pengujian	TP	TN	FN	FP	Akurasi
1	165499	0	0	0	100%
2	161299	0	0	0	100%
3	162175	0	0	0	100%
4	168873	0	0	0	100%
5	166353	0	0	0	100%
Rata-rata					100%

6.2.2.6 Hasil Pengujian Akurasi

Hasil rata-rata yang didapatkan untuk pengujian ini adalah 100%. Hasil dari akurasi scapy dapat dilihat pada Tabel 6.39.

Tabel 6.39 Rata-rata Akurasi Scapy

Kategori	Rata-rata Akurasi
NORMAL	100%
SYN *FLOOD* IPv4	100%
SYN *FLOOD* IPv6	100%
UDP *FLOOD* IPv4	100%
UDP *FLOOD* IPv6	100%
Rata-rata	100%

6.2.3 Pengujian Penanggulangan Serangan IDS

Berdasarkan hasil pengujian yang dilakukan, penanggulangan serangan dapat dilakukan ketika serangan tersebut berhasil dideteksi. Sehingga fungsionalitas IDS untuk melakukan block pada penyerang berhasil dilakukan. Adapun hasil yang didapatkan pada pengujian penanggulangan dapat dilihat pada subbab pada bagian ini.

6.2.3.1 Pengujian Trafik Serangan UDP Flood IPv4

Trafik Serangan UDP *Flood* IPv4 adalah trafik yang ada pada lingkungan penelitian ketika terjadi serangan UDP *flood* IPv4. Pengujian penanggulangan serangan dilakukan untuk mengetahui keberhasilan IDS dalam menanggulangi serangan. Hasil dari proses pengujian ini dapat dilhat pada Tabel 6.40.

Tabel 6.40 Pengujian Trafik Serangan UDP *Flood* IPv4

Kode	PS_01
Pengujian	Trafik Serangan UDP *Flood* IPv4
Tujuan	Dapat mengetahui keberhasilan menanggulangi serangan pada IDS ketika terjadi serangan
Prosedur	1. IDS dijalankan 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv4 paket 3. Melakukan cek *iptable*s pada sistem operasi
Hasil Yang diharapkan	IDS dapat memblokir penyerang ketika terjadi serangan UDP *flood* IPv4
Hasil Pengujian	IDS dapat memblokir penyerang ketika terjadi serangan UDP *flood* IPv4

Gambar 6.13 menunjukkan hasil dari *firewall rules* dari serangan UDP IPv4. Serangan ini akan ditangulangi dengan menambahkan *rule* pada *firewall* dengan menggunakan aplikasi *iptables*. *Firewall* akan melakukan drop packet *input, output* dan *forward* kepada penyerang berdasarkan alamat IP.

```
pi@masterB:~/raspyids $ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  192.168.42.28        anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  192.168.42.28        anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  192.168.42.28        anywhere
```

Gambar 6.13 Hasil *Iptable*s pada Serangan UDP IPv4

6.2.3.2 Pengujian Trafik Serangan UDP Flood IPv6

Trafik Serangan UDP *Flood* IPv6 adalah trafik yang ada pada lingkungan penelitian ketika terjadi serangan UDP *flood* IPv6. Pengujian penanggulangan serangan dilakukan untuk mengetahui keberhasilan IDS dalam menanggulangi serangan. Hasil dari proses pengujian ini dapat dilhat pada Tabel 6.41.

Tabel 6.41 Pengujian Trafik Serangan UDP *Flood* IPv6

Kode	PS_02
Pengujian	Trafik Serangan UDP *Flood* IPv6
Tujuan	Dapat mengetahui keberhasilan menanggulangi serangan pada IDS ketika terjadi serangan
Prosedur	1. IDS dijalankan 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv6 paket 3. Melakukan cek *iptable*s pada sistem operasi
Hasil Yang diharapkan	IDS dapat memblokir penyerang ketika terjadi serangan UDP *flood* IPv6
Hasil Pengujian	IDS dapat memblokir penyerang ketika terjadi serangan UDP *flood* IPv6

Gambar 6.14 menunjukkan hasil dari *firewall rules* dari serangan UDP IPv6. Serangan ini akan ditangulangi dengan menambahkan *rule* pada *firewall* dengan

menggunakan aplikasi *iptables*. *Firewall* akan melakukan drop packet *input, output* dan *forward* kepada penyerang berdasarkan alamat IP.

```
pi@masterB:~/raspyids $ sudo ip6tables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      fe80::8616:f9ff:fe08:e322  anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all      fe80::8616:f9ff:fe08:e322  anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      fe80::8616:f9ff:fe08:e322  anywhere
```

Gambar 6.14 Hasil Ip6tables pada Serangan UDP IPv6

6.2.3.3 Pengujian Trafik Serangan SYN Flood IPv4

Trafik Serangan SYN *Flood* IPv4 adalah trafik yang ada pada lingkungan penelitian ketika terjadi serangan SYN *flood* IPv4. Pengujian penanggulangan serangan dilakukan untuk mengetahui keberhasilan IDS dalam menanggulangi serangan. Hasil dari proses pengujian ini dapat dilhat pada Tabel 6.42.

Tabel 6.42 Pengujian Trafik Serangan SYN *Flood* IPv4

Kode	PS_03
Pengujian	Trafik Serangan SYN *Flood* IPv4
Tujuan	Dapat mengetahui keberhasilan menanggulangi serangan pada IDS ketika terjadi serangan
Prosedur	1. IDS dijalankan 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 SYN IPv4 paket 3. Melakukan cek *iptables* pada sistem operasi
Hasil Yang diharapkan	IDS dapat memblokir penyerang ketika terjadi serangan SYN *flood* IPv4
Hasil Pengujian	IDS dapat memblokir penyerang ketika terjadi serangan SYN *flood* IPv4

Gambar 6.15 menunjukkan hasil dari *firewall rules* dari serangan SYN IPv4. Serangan ini akan ditangulangi dengan menambahkan *rule* pada *firewall* dengan menggunakan aplikasi *iptables*. *Firewall* akan melakukan drop packet *input, output* dan *forward* kepada penyerang berdasarkan alamat IP.

```
pi@masterB:~/raspyids $ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  192.168.42.28        anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  192.168.42.28        anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  192.168.42.28        anywhere
```

Gambar 6.15 Hasil *Iptables* pada Serangan TCP IPv4

6.2.3.4 Pengujian Trafik Serangan SYN Flood IPv6

Trafik Serangan SYN *Flood* IPv6 adalah trafik yang ada pada lingkungan penelitian ketika terjadi serangan SYN *flood* IPv6. Pengujian penanggulangan serangan dilakukan untuk mengetahui keberhasilan IDS dalam menanggulangi serangan. Hasil dari proses pengujian ini dapat dilhat pada Tabel 6.43.

Tabel 6.43 Pengujian Trafik Serangan SYN *Flood* IPv6

Kode	PS_04
Pengujian	Trafik Serangan SYN *Flood* IPv6
Tujuan	Dapat mengetahui keberhasilan menanggulangi serangan pada IDS ketika terjadi serangan
Prosedur	4. IDS dijalankan 5. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 SYN IPv6 paket 6. Melakukan cek *iptable*s pada sistem operasi
Hasil Yang diharapkan	IDS dapat memblokir penyerang ketika terjadi serangan SYN *flood* IPv6
Hasil Pengujian	IDS dapat memblokir penyerang ketika terjadi serangan SYN *flood* IPv6

Gambar 6.16 menunjukkan hasil dari *firewall rules* dari serangan SYN IPv6. Serangan ini akan ditangulangi dengan menambahkan *rule* pada *firewall* dengan menggunakan aplikasi *iptables*. *Firewall* akan melakukan drop packet *input, output* dan *forward* kepada penyerang berdasarkan alamat IP.

```
pi@masterB:~/raspyids $ sudo ip6tables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      fe80::8616:f9ff:fe08:e322  anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all      fe80::8616:f9ff:fe08:e322  anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all      fe80::8616:f9ff:fe08:e322  anywhere
```

Gambar 6.16 Hasil Ip6tables pada Serangan TCP IPv6

6.2.3.5 Hasil Pengujian Penanggulangan Serangan

Hasil pengujian yang didapatkan pada pengujian ini adalah berhasil menanggulangi semua serangan yang diuji. Sehingga fungsionalitas IDS penanggulangan serangan dapat dikatakan mampu menanggulangi serangan. Hasil pengujian dapat dilihat pada Tabel 6.44.

Tabel 6.44 Hasil Pengujian Penanggulangan Serangan

Kategori	Status Hasil Pengujian
SYN *FLOOD* IPv4	Valid
SYN *FLOOD* IPv6	Valid
UDP *FLOOD* IPv4	Valid
UDP *FLOOD* IPv6	Valid

6.2.4 Pengujian Akurasi *Alert* IDS

6.2.4.1 Pengujian Kemampuan Menampilkan *Alert* Terhadap Serangan UDP *flood* IPv4

Pengujian ini dilakukan untuk mengukur kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv4 dari implementasi IDS pada *middleware* ketika trafik sedang normal dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.45.

Tabel 6.45 Pengujian kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv4

Kode	AA_01
Pengujian	kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv4
Tujuan	Mengetahui kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv4
Prosedur	1. IDS dijalankan dan output aplikasi disimpan dalam sebuah file 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv4 paket 3. Melakukan melakukan perhitungan jumlah *alert* yang diberikan dari file tersebut.
Hasil Yang diharapkan	Dapat mengetahui rata-rata kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv4
Hasil Pengujian	Mengetahui rata-rata kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv4

Pengujian ini dilakukan dengan merekam kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv4 dan IDS berjalan. Hasil kecepatan klasifikasi dapat dilihat pada Tabel 6.46.

Tabel 6.46 Pengujian kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv4

Pengujian	TP	TN	FN	FP	Akurasi
1	223	777	0	0	22,30%
2	176	824	0	0	17,60%
3	168	832	0	0	16,80%
4	184	816	0	0	18,40%
5	171	829	0	0	17,10%
Rata-rata					18,44%

6.2.4.2 Pengujian Kemampuan Menampilkan *Alert* Terhadap Serangan *UDP flood IPv6*

Pengujian ini dilakukan untuk mengukur kemampuan menampilkan *alert* dari implementasi IDS pada *middleware* ketika trafik terjadi serangan UDP *flood* IPv6 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.47.

Tabel 6.47 Pengujian kemampuan menampilkan *alert* terhadap serangan UDP *flood IPv6*

Kode	AA_02
Pengujian	Kemampuan menampilkan *alert* Serangan UDP *flood* IPv6
Tujuan	Mengetahui kemampuan menampilkan *alert* terhadap Serangan UDP *flood* IPv6
Prosedur	1. IDS dijalankan dan output aplikasi disimpan dalam sebuah file 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 UDP IPv6 paket 3. Melakukan melakukan perhitungan jumlah *alert* yang diberikan dari file tersebut.
Hasil Yang diharapkan	Dapat mengetahui rata-rata kemampuan menampilkan *alert* terhadap serangan UDP *flood* IPv6
Hasil Pengujian	Mengetahui rata-rata kemampuan menampilkan Albert terhadap serangan UDP *flood* IPv6

Pengujian ini dilakukan dengan merekam kemampuan menampilkan *alert* ketika kondisi serangan UDP *flood IPv6* dan IDS berjalan. Hasil kecepatan klasifikasi dapat dilihat pada Tabel 6.48.

Tabel 6.48 Pengujian kemampuan menampilkan *alert* terhadap Serangan UDP *flood* Ipv6

Pengujian	TP	TN	FN	FP	Akurasi
1	196	777	0	0	19,60%
2	177	824	0	0	17,70%
3	165	832	0	0	16,50%
4	170	816	0	0	17,00%
5	180	829	0	0	18,00%
Rata-rata					17,76%

6.2.4.3 Pengujian kemampuan menampilkan *alert* terhadap Serangan *SYN flood IPv4*

Pengujian ini dilakukan untuk mengukur kemampuan menampilkan *alert* dari implementasi IDS pada *middleware* ketika trafik terjadi serangan SYN *flood* IPv4 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.49.

Tabel 6.49 Pengujian akurasi *detection engine* pada serangan SYN *flood* IPv4

Kode	AA_03
Pengujian	Kemampuan menampilkan *alert* terhadap Serangan SYN *flood* IPv4
Tujuan	Mengetahui kemampuan menampilkan Albert terhadap Serangan SYN *flood* IPv4
Prosedur	1. IDS dijalankan dan output aplikasi disimpan dalam sebuah file 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 SYN IPv4 paket 3. Melakukan melakukan perhitungan jumlah *alert* yang diberikan dari file tersebut.

Hasil Yang diharapkan	Dapat mengetahui rata-rata kemampuan menampilkan *alert* terhadap Serangan SYN *flood* IPv4
Hasil Pengujian	Mengetahui rata-rata kemampuan menampilkan *alert* terhadap Serangan SYN *flood* IPv4

Pengujian ini dilakukan dengan merekam kemampuan menampilkan *alert* ketika kondisi serangan SYN *flood Ipv4* dan IDS berjalan. Hasil kecepatan klasifikasi dapat dilihat pada Tabel 6.50.

Tabel 6.50 Pengujian kemampuan menampilkan *alert* terhadap Serangan SYN *flood* Ipv4

Pengujian	TP	TN	FN	FP	Akurasi
1	176	824	0	0	17,60%
2	173	827	0	0	17,30%
3	160	840	0	0	16,00%
4	172	828	0	0	17,20%
5	173	827	0	0	17,30%
Rata-rata					17,08%

6.2.4.4 Pengujian Kemampuan Menampilkan *Alert* Terhadap Serangan *SYN flood IPv6*

Pengujian ini dilakukan untuk mengukur kemampuan menampilkan Albert dari implementasi IDS pada *middleware* ketika trafik terjadi serangan SYN *flood* IPv6 dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.51.

Tabel 6.51 Pengujian kemampuan menampilkan *alert* terhadap serangan SYN *flood* IPv6

Kode	AA_04
Pengujian	Kemampuan menampilkan *alert* terhadap Serangan SYN *flood* IPv6
Tujuan	Mengetahui kemampuan menampilkan *alert* terhadap Serangan SYN *flood* IPv6
Prosedur	1. IDS dijalankan dan output aplikasi disimpan dalam sebuah file 2. Penyerang melakukan penyerangan kepada *middleware* dengan mengirimkan 1000 SYN IPv6 paket 3. Melakukan melakukan perhitungan jumlah *alert* yang diberikan dari file tersebut.
Hasil Yang diharapkan	Dapat mengetahui rata-rata kemampuan menampilkan *alert* terhadap Serangan SYN *flood* IPv6
Hasil Pengujian	Mengetahui rata-rata kemampuan menampilkan *alert* terhadap Serangan SYN *flood* IPv6

Pengujian ini dilakukan dengan merekam kemampuan menampilkan *alert* ketika kondisi serangan SYN *flood Ipv6* dan IDS berjalan. Hasil kecepatan klasifikasi dapat dilihat pada Tabel 6.52.

Tabel 6.52 Pengujian kemampuan menampilkan *alert* terhadap Serangan SYN *flood* Ipv6

Pengujian	TP	TN	FN	FP	Akurasi
1	156	844	0	0	17,60%
2	162	838	0	0	17,30%
3	166	834	0	0	16,00%
4	164	836	0	0	17,20%
5	171	829	0	0	17,30%
Rata-rata					17,08%

6.2.4.5 Hasil Pengujian Akurasi Pesan Peringatan

Hasil rata-rata yang didapatkan untuk pengujian ini adalah 17,42%. Hasil dari akurasi scapy dapat dilihat pada Tabel 6.53.

Tabel 6.53 Rata-rata Akurasi Pesan Peringatan Scapy

Kategori	**Rata-rata Akurasi**
SYN *FLOOD* IPv4	17,08%
SYN *FLOOD* IPv6	16,38%
UDP *FLOOD* IPv4	18,44%
UDP *FLOOD* IPv6	17,76%
Rata-rata	17,42%

6.2.5 Pengujian Logging

Pengujian Logging dilakukan untuk mengetahui apakah IDS mampu membuat log file ketika IDS dapat berjalan. Hasil pengujian logging yang didapatkan adalah aplikasi IDS mampu melakukan log aktifitas ketika terjadi serangan dan ketika aplikasi IDS dimatikan. Hasil pengujian ini dapat dilihat pada subbab pada bagian ini.

6.2.5.1 Pengujian Kemampuan Logging

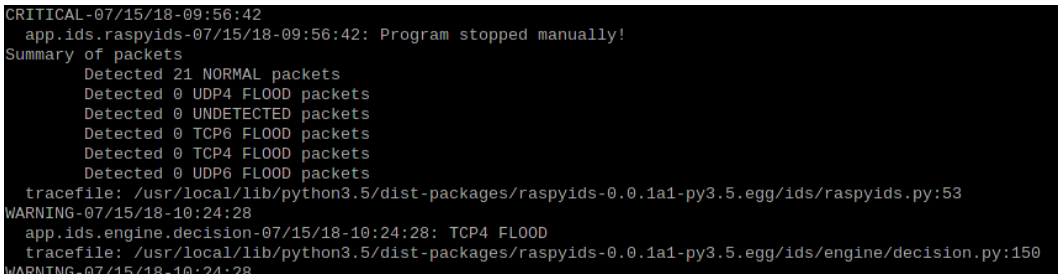
Pada Tabel 6.54 menunjukkan hasil dari pengujian LG_01. Pengujian ini akan mengecek file log yang dibuat oleh IDS.

Tabel 6.54 Pengujian Kemampuan Logging

Kode	LG_01
Pengujian	Pengujian Kemampuan Logging
Tujuan	Mengetahui kinerja logging pada ids
Prosedur	1. Melakukan pengecekan terhadap log file selama menjalankan aplikasi IDS.
Hasil Yang diharapkan	Dapat memberikan logging ketika menjalankan IDS
Hasil Pengujian	File logging didapatkan ketika menjalankan IDS

Fitur logging yang diuji pada sistem ini berhasil. Log file yang diperoleh berupa hasil klasifikasi ketika ids tesebut dimatikan. Selain itu, file log juga berisi *alert*

serangan berupa waktu dan jenis serangan. Gambar 6.17 memperlihatkan hasil dari logging pada IDS.



Gambar 6.17 Pengujian Logging

6.2.5.2 Hasil Pengujian Akurasi

Hasil pengujian yang didapatkan pada pengujian ini adalah berhasil melakukan *logging*. Hasil pengujian dapat dilihat pada Tabel 6.55.

Tabel 6.55 Hasil Pengujian Logging

Kategori	Status Hasil Pengujian
Pengujian Logging	Valid

6.2.6 Pengujian *Availability* IDS

6.2.6.1 Pengujian Sistem Dapat Berjalan Lebih Dari 1 Jam

Pengujian ini dilakukan untuk mengukur apakah benar sistem dapat berjalan lebih dari 1 jam dari implementasi IDS pada *middleware* ketika trafik sedang normal dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.56.

Tabel 6.56 Pengujian sistem dapat berjalan lebih dari 1 jam

Kode	AV_01
Pengujian	Sistem Dapat Berjalan Lebih Dari 1 Jam
Tujuan	Mengetahui apakah sistem dapat berjalan lebih dari 1 jam
Prosedur	1. Menjalankan IDS sebagai sebuah service 2. Mengecek status IDS keesokan harinya
Hasil Yang diharapkan	Dapat mengetahui bahwa sistem dapat berjalan lebih dari 1 jam
Hasil Pengujian	Mengetahui bahwa sistem dapat berjalan lebih dari 1 jam

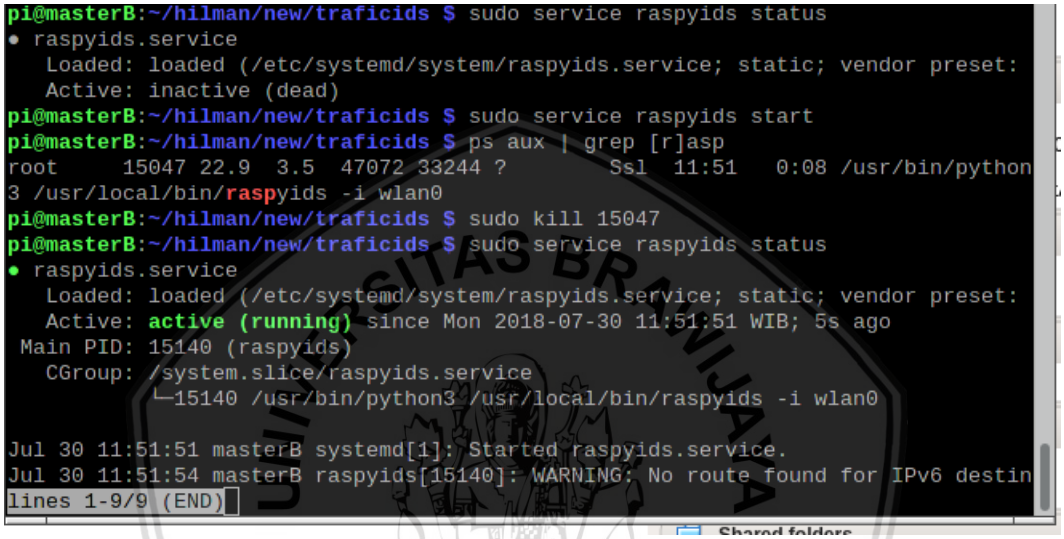
6.2.6.2 Pengujian *Auto Restart*

Pengujian ini dilakukan untuk mengetahui apakah sistem mampu melakukan *auto restart* dari implementasi IDS pada *middleware* dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.57.

Tabel 6.57 Pengujian *auto restart*

Kode	AV_02

Pengujian	*Auto Restart*
Tujuan	Mengetahui apakah sistem mampu melakukan *auto restart*
Prosedur	1. Menjalankan IDS sebagai sebuah service 2. Mematikan paksa IDS 3. Mengecek status IDS.
Hasil Yang diharapkan	Dapat mengetahui bahwa sistem mampu melakukan *auto restart*
Hasil Pengujian	Mengetahui bahwa sistem mampu melakukan *auto restart*



Gambar 6.18 Pengujian Logging

6.2.6.3 Hasil Pengujian Availability

Hasil pengujian yang didapatkan pada pengujian ini adalah berhasil melakukan pengujian a*vailability*. Hasil pengujian dapat dilihat pada Tabel 6.58.

Tabel 6.58 Hasil Pengujian *Availability*

Kategori	Status Hasil Pengujian
Sistem Dapat Berjalan Lebih Dari 1 Jam	Valid
Auto Restart	Valid

6.2.7 Pengujian *Performance* IDS

6.2.7.1 Pengujian Mengklasifikasi Paket Kurang Dari Satu Detik

Pengujian ini dilakukan untuk mengklasifikasi paket kurang dari satu detik dari implementasi IDS pada *middleware* ketika trafik sedang normal dengan IDS. Metode pengujian dapat dilihat pada Tabel 6.59.

Tabel 6.59 Pengujian mengklasifikasi paket kurang dari satu detik

Kode	PF_01
Pengujian	Sistem Dapat Mengklasifikasi Paket Kurang Dari Satu Detik

Tujuan	Mengetahui apakah sistem dapat mengklasifikasi paket kurang dari satu detik
Prosedur	1. Menyimpan waktu ketika paket masuk dalam detection engine dan waktu ketika selesai diklasifikasi 2. Menghitung waktu rata-rata yang dibutuhkan dalam 5000 paket yang diterima
Hasil Yang diharapkan	Dapat mengetahui bahwa sistem dapat mengklasifikasi paket kurang dari satu detik
Hasil Pengujian	Mengetahui bahwa sistem dapat mengklasifikasi paket kurang dari satu detik

Pada pengujian pada bab 6.1.3 menyatakan bahwa rata-rata kecepatan melakukan klasifikasi paket adalah 0,030629 detik. Sehingga fungsionalitas ini terpenuhi.

6.2.7.2 Hasil Pengujian Performance

Hasil pengujian yang didapatkan pada pengujian ini adalah berhasil melakukan pengujian *performance*. Hasil pengujian dapat dilihat pada Tabel 6.60.

Tabel 6.60 Hasil Pengujian *Performance*

Kategori	**Status Hasil Pengujian**
Sistem Dapat Mengklasifikasi Paket Kurang Dari Satu Detik	Valid